View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Digital library of Brno University of Technology

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# SEMIANALYTICKÝ VÝPOČET KOEFICIENTŮ FOURIEROVY ŘADY

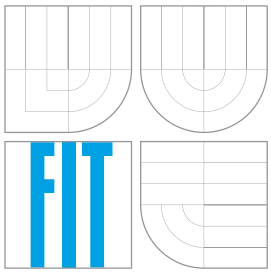
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ HÉGR

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# SEMIANALYTICKÝ VÝPOČET KOEFICIENTŮ FOURIEROVY ŘADY

SEMIANALYTICAL COMPUTATION OF FOURIER SERIES COEFICIENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ HÉGR

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÁCLAV ŠÁTEK, Ph.D.

BRNO 2012

# Abstrakt

Tato bakalářská práce se zabývá semianalytickým výpočtem koeficientů Fourierovy řady. Cílem práce je vytvořit uživatelské rozhraní pro program TKSL/C. Obsahuje teorii Fourierovy řady a Fourierovy transformace. V další kapitole jsou popsány již existující programy. Dále je navržena vlastní implemetace rozhraní, jsou uvedeny příklady aplikace a shrnuty reakce od uživatelů.



# Abstract

This bachelor thesis deals with semianalytical computation of the coefficients of the Fourier series. The aim of this work is to create a graphical user interface for program TKSL/C. The work contains the theory of Fourier series and Fourier transforms. The state of the art software is described in the following chapter. In the next parts the thesis describes the concept of the implementation of the user interface, some tutorial examples which were used and the summary of user testing.



# Klíčová slova

Fourierova řada, Fourierova transformace, DFT, TKSL/C, TKSL/386, MATLAB, diferenciální rovnice, grafické uživatelské rozhraní, C++, Qt.

# Keywords

Fourier series, Fourier transform, DFT, TKSL/C, TKSL/386, MATLAB, diffrential equations, graphical user interface, C++, Qt.


# Citace

Tomáš Hégr: Semianalytický výpočet koeficientů
Fourierovy řady, bakalářská práce, Brno, FIT VUT v Brně, 2012

# Semianalytický výpočet koeficientů Fourierovy řady

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana inženýra Václava Šátka.

.....
Tomáš Hégr
14. května 2012

## Poděkování

Na tomto místě bych rád poděkoval za odbornou pomoc a vedení Ing. Václavu Šátkovi, dále pak autorovi TKSL/C panu Ing. Ondřejovi Holubovi za konzultace a úpravu programu a na závěr své rodině a přátelům za podporu.

© Tomáš Hégr, 2012.
*Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.*

# Obsah

# Kapitola 1

# Úvod

Problematika Fourierovy transformace má široký význam uplatnění ve spojení s počítačovou technikou. Jedná se o převod funkcí na goniometické řady, kterého se využívá při zpracování signálů, ale také v několika oborech fyziky. Každé z odvětví ale klade jiné nároky na přesnost a rychlost výpočtů. Například pro zpracování signálu jsou spíše důležité výsledky v co nejkratším čase, zatímco astrofyzika vyžaduje co nejpřesnější výsledky.

Řešení těchto problémů (úloh) lze samozřejmě většinou nalézt i analyticky (s využitím různých numerických metod), nicméně vývoj ukazuje, že mnohem výhodnější a efektivnější je využívat speciální programy pro výpočet. V tomto projektu se budu zabývat především programem TKSL/386 a jeho novější verzí CLTKSL, které řeší integrály převodem na diferenciální rovnice. Navíc je nutné si uvědomit, že výsledky dostáváme ve dvou rovinách, kde o komplexní oblast se zajímáme především při studiu signálů, oproti tomu výsledky z reálné oblasti jsou potřebné například v astrofyzice.

Pro větší přehlednost a možnost pozdějších úprav je celá práce vysázena pomocí značkovacího jazyka LaTeX. S využitím sazeb vzorců a tabulek má práce mnohem čitelnější formát. Při vytvážení zdrojového textu zprávy jsem využíval knihu pana Kopka ([3]) a informace ze školních webových stránek ([10]).

## 1.1 Cíl práce

Cílem této práce bylo seznání s problematikou semianalytického výpočtu koeficientů Fourierovy řady s využitím dostupného programového vybavení v této oblasti (programy TKSL/386, TKSL/C, MATLAB a Maple). Dalším úkolem bylo vytvořit popis parametrů a implementace uživatelského rozhraní pro CLTKSL, které doposud běží pouze v řežimu příkazové řádky.

Vytvořené uživatelské rozhraní si za cíl pokládá zpřístupnění programu pro větší skupinu uživatelů (není nutná znalost příkazů a většinu nastavení můžeme realizovat pomocí vytvořeného rozhraní). Tím pádem se vyplynul jeden z požadavků na multiplatformovost, který zároveň měl za úkol prověřit možnost šíření TKSL/C ve svetě UNIXových distribucí.

Práce je rozděle do 8 logických celků, které tvoří jednotlivé kapitoly.
**První kapitola** představuje tento úvod.
**Druhá kapitola** obsahuje základní poznatky o Fourierově řadě a Fourierově transformaci.
**Třetí kapitola** se zabývá vyřešením dvou demonstračních příkladů na rozklad funkce na goniometrickou řadu.
**Čtvrtá kapitola** je zaměřená na program TKSL a jeho inovovanou verzi TKSL/C.

**Pátá kapitola** porovnává existující programové vybavení pro výpočet Fouerierovy transformace (zaměřuje se na MATLAB, Maple a TKSL).
**Šestá kapitola** pojednává o mnou navrženém a naimplementovaném grafickém uživatelském rozhraní pro CLTKSL.
**Sedmá kapitola** se zabývá hodnocením rozhraní z předchozí kapitoly na základě dotazníků od uživatelů.
**Osmá kapitola** shrnuje výsledky a uzavírá práci. Naznačuje také další možné směry vývoje vytvořené aplikace.

# Kapitola 2

# Teoretický rozbor

Informace uvedené v této kapitole jsem čerpal z [1, 2, 5, 6]

## 2.1 Fourierova řada

Fourierova řada slouží k zápisu jakéhokoliv periodického průběhu za pomoci posloupnosti goniometrických funkcí sinus a kosinus. Běžně zobrazuje pouze jednu jeho periodu (nicméně není vyloučeno užití pro více period). Rozkládáme funkci do trigonometrických řad, které pokrývají právě jednu periodu zkoumané křivky.

Pomocí této řady lze rozložit i často velmi komplikované funkce, které by jinak byl problém zobrazit. Lze tak studovat průběh funkcí, které nejsou spojité (nanejvýš na množině míry nula) a může mít i nespojitou derivaci (tj. funkce může mít ostré hrany).

O zkoumaných periodických křivkách víme, že pro ně platí linearita, možnost posunu v čase a změna časového měřítka. To znázorňuje následující tabulka:

| | $x(t)$ | $c_k$ |
|---|---|---|
| linearita | $a * x_a(t) + n * x_b(t)$ | $a * c_{a,k} + b * c_{b,k}$ |
| posunutí v čase | $x(t - \tau)$ | $c_k * e^{-jk\omega\tau}$ |
| změna časového měřítka | $x(mt)$ | $c_k$ |

## 2.2 Reálný tvar Fourierovy řady

Fourierova řada rozloží periodickou funkci na trigoniometrickou, neboli funkční řadu. V reálné oblasti se využívá následující tvar:

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos kt + b_k \sin kt) \tag{2.1}$$
$$f(t) = \frac{a_0}{2} + a_1 \cos \omega t + b_1 \sin \omega t + a_2 \cos \omega t + b_2 \sin \omega t + \dots$$

kde $a_k$ a $b_k$ jsou konstanty.

Částečný součet řady lze vyjádřit jako:

$$S_n(t) = \frac{a_0}{2} + \sum_{k=1}^{n}(a_n \cos \omega t + b_n \sin \omega t) \tag{2.2}$$

Pokud trigoniometrická řada konverguje, pak lze koeficienty Fourierovy řady vyjádřit jako:

$$a_0 = \frac{1}{T} * \int_0^{2T} f(t)dt \tag{2.3}$$

$$a_k = \frac{1}{T} * \int_0^{2T} f(t) * cos(k * t)dt \tag{2.4}$$

$$b_k = \frac{1}{T} * \int_0^{2T} f(t) * sin(k * t)dt \tag{2.5}$$

kde $k \in Z$, $k > 0$ a $T$ je perioda.

### 2.2.1 Užití reálného tvaru Fourierovy řady

Tento tvar FŘ má význam například při řešení jednodušších rovnic, kdy lze pomocí analytického řešení ověřit správnost rozkladu. Jako příklad lze uvést rozklad rovnice

$$f(t) = \frac{\cos 2t}{\sin t + \cos t} \tag{2.6}$$

Je evidentní, že v zadané rovnici (2.6) je $\omega = 1rad/s$.

Pro rozklad použijeme následující vztahy:

$$\cos 2t = \cos^2 t - \sin^2 t$$
$$a^2 - b^2 = (a - b) * (a + b)$$

Postupujeme následovně:

$$\frac{\cos 2t}{\sin t + \cos t} = \frac{\cos^2 t - \sin^2 t}{\sin t + \cos t} = \frac{(\cos t - \sin t) * (\cos t + \sin t)}{\sin t + \cos t} =$$
$$= \cos t - \sin t = 1 * \cos t + (-1) * \sin t$$

Určil jsem tedy koeficienty FŘ:

$$a_1 = 1$$
$$b_1 = -1$$
ostatní koeficienty jsou rovny nule

Ověření tohoto výpočtu pomocí TKSL/C je v souboru pr1.tksl, který je součástí zdrojových kódů vytvořeného grafického rozhraní.

## 2.3 Komplexní tvar Fourierovy řady

Pro využití v oblasti signálů se často využívají komplexní funkce. Průběh funkce se vyobrazuje v čase, a proto se nahrazuje obecná proměnná $x$ za čas $t$. Fourierova řada představuje sumu komplexních exponenciál (těmi lze vyjádřit libovolnou funkci cosinus a z toho důvodu se snažíme libovolný periodický signál rozložit do řady komplexních exponenciál). Periodický signál $x(t) = x(t + T_1)$, kde $T_1$ je základní perioda, rozložíme na tvar:

$$x(t) = \sum_{k=-\infty}^{\infty} c_k * e^{jk\omega_1 t} \tag{2.7}$$

kde:

$\omega_1 = \frac{2\pi}{T}$ je základní úhlový kmitočet signálu,
$e^{jk\omega_1 t}$ pro $k \in Z$ nazýváme vztažené komplexní exponenciály,
$c_k$ jsou komplexní koeficienty Fourierovy řady.

Pro reálné signály $x(t)$ budou koeficienty $c_k$ a $c_{-k}$ komplexně sdružené. Koeficient $c_0$ je sdružený sám se sebou (jelikož se jedná o stejnosměrnou složku signálu). Pro reálné signály se dá tedy rovnice napsat i takto:

$$s(t) = c_0 + \sum_{k=1}^{\infty}(c_k * e^{jk\omega_1 t} + c_{-k} * e^{-jk\Omega_1 t})$$
$$s(t) = c_0 + \sum_{k=1}^{\infty}[c_k * \cos(k\omega_1 t + \varphi k)]$$

Koeficienty Fourierovy řady v komplexním tvaru lze určit pomocí vztahu:

$$c_k = \frac{1}{T}\int_{T_1} x(t) * e^{-jk\omega_1 t} dt \tag{2.8}$$

kde

$\int_{T_1}$ značí integraci přes libovolnou periodu.

### 2.3.1 Užití komplexního tvaru Fourierovy řady

Pomocí Fourierovy řady se snažíme rozložit libovolný periodický signál do řady komplexních exponenciál. Díky nim lze elegantně vyjádřit křivku cosinu s libovolnou fází. Toto tvrzení vychází ze vzorce:

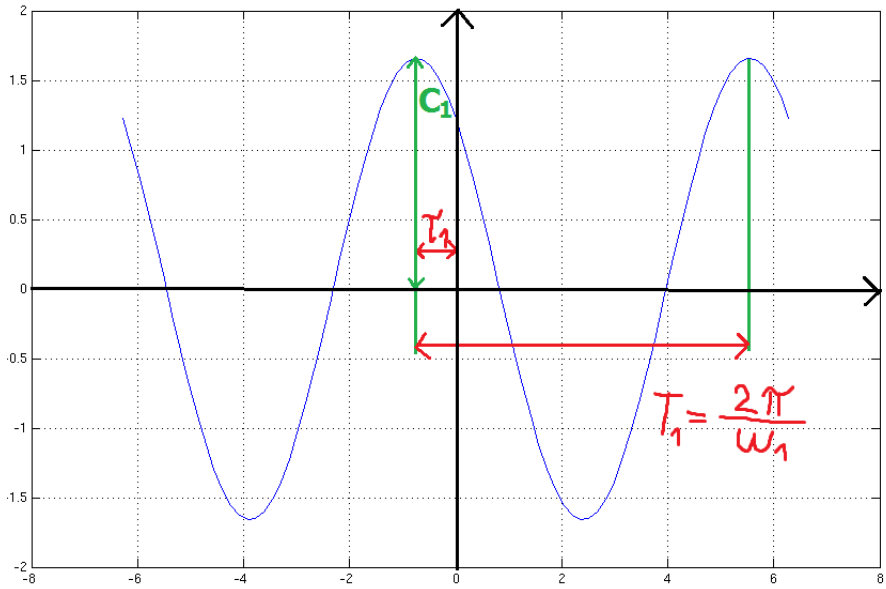
$$\cos(x) = \frac{e^{jx} + e^{-jx}}{2} \tag{2.9}$$

kde $j$ je komplexní jednotka. Z tohoto vztahu lze odvodit vyjádření obecného cosinu:

$$C_1 * \cos(\omega_1 t + \varphi_1) = \frac{C_1}{2} * e^{j(\omega_1 t + \varphi_1)} + \frac{C_1}{2} * e^{-j(\omega_1 t + \varphi_1)} =$$
$$= \frac{C_1}{2} * e^{j\varphi_1} * e^{j\omega_1 t} + \frac{C_1}{2} * e^{-j\varphi_1} * e^{-j\omega_1 t}$$

kde

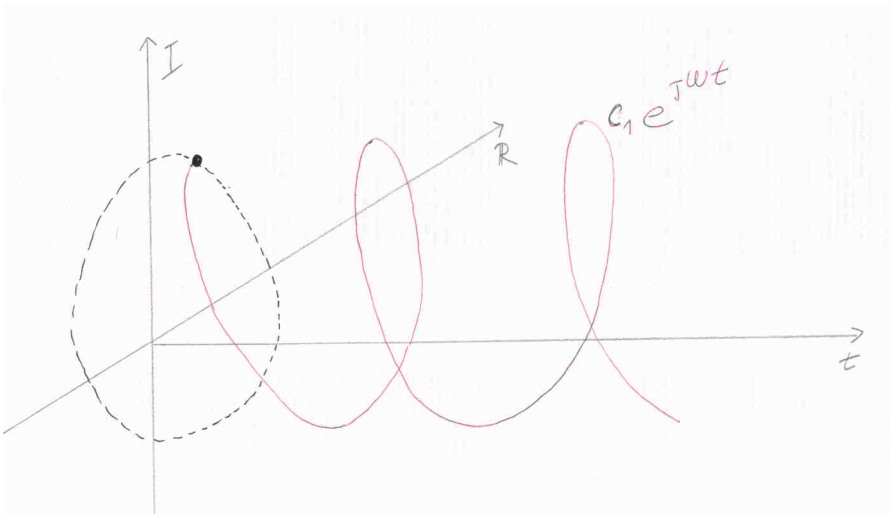
$$C_1 = \frac{C_1}{2} * e^{j\varphi_1}$$
a
$$C_{-1} = \frac{C_1}{2} * e^{-j\varphi_1}$$




Obrázek 2.1: Průmět komplexních exponenciál.

jsou komplexní exponenciály. Jejich průměr znázorňuje například graf 2.1.

Funkce představuje dvě komplexní exponenciály (spirály) v prostoru. Dle následujícího obrázku (2.2) je vidět, že poměr představuje $C_1$ a $\varphi_1$ posun na ose x.




Obrázek 2.2: Hodnoty koeficientů $a_k$ a $b_k$

### 2.3.2 Převod základních funkcí na Fourierovu řadu

Při některých výpočtech je vhodné si výsledky porovnat se známým řešením. Proto existuje několik různých převodních tabulek, pomocí kterých lze převést výsledky z reálného tvaru do komplexní oblasti a samozřejmě i naopak. Zde uvádím tabulku převzatou z [2], ze které

využijeme vzorce v dalších kapitolách.

| | $\lvert c_k\rvert$, arg $c_k$ | $a_k$, $\varphi_k$ | $a_k$, $b_k$ |
|---|---|---|---|
| $\lvert c_k\rvert$, arg $c_k$ | | $A_0 = c_0$<br>$A_k = 2\lvert c_k\rvert$<br>$\varphi_k =$arg $c_k$ | $a_0 = c_0$<br>$a_k = 2Re \cdot c_k$<br>$b_k = -2Im \cdot c_k$ |
| $a_k$, $\varphi_k$ | $c_0 = A_0$<br>$\lvert c_k\rvert = \frac{1}{2}A_k$<br>arg $c_k = \varphi_k$ | | $a_0 = A = 0$<br>$a_k = A_k * \cos\varphi_k$<br>$b_k = -A_k * \sin\varphi_k$ |
| $a_k$, $b_k$ | $c_0 = a_0$<br>$c_k = \frac{1}{2}\sqrt{(a_k^2 + b_k^2)}$<br>$\cos\varphi_k = \frac{a_k}{2\lvert c_k\rvert}$<br>$\sin\varphi_k = \frac{-b_k}{2\lvert c_k\rvert}$<br>$c_k = \frac{1}{2}a_k - j\frac{1}{2}b_k$ | $A_0 = a_0$<br>$A_k = \sqrt{(a_k^2 + b_k^2)}$<br>$\cos\varphi_k = \frac{a_k}{A_k}$<br>$\sin\varphi_k = \frac{-b_k}{A_k}$ | |

## 2.4 Fourierova transformace

Fourierovou transformací rozumíme proces převodu funkce na součet členů Fourierovy řady. Jedná se tudíž o nalezení koeficientů $a_0, a_1, \ldots b_n$. Takto lze získat pouze funkce splňující podmínku, že jsou spojité a periodické (alespoň po částech).

Protože je Fourierova řada omezená pouze na periodické signály (rozkládáme ji na součet trigonometrických funkcí sinus a cosinus), lze získat jejím omezením spektrální funkci pro neperiodické signály. Tuto funkci nazveme Fouriérův obraz nebo jen obraz funkce $x(t)$. Značíme ji $X(j\omega)$ a určíme jako:

$$X(j\omega) = \int_{-\infty}^{\infty} x(t) * e^{-j\omega t} dt \tag{2.10}$$

Vyjdeme-li ze syntézy signálů koeficientů Fourierovy řady, lze určit zpětnou Fourierovu transformaci jako:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) * e^{j\omega t} d\omega \tag{2.11}$$

Fourierova transformace se vyznačuje několika základními vlastnostmi. U Fourierovy řady například nemůžeme uvažovat Diracův impuls, protože není v základní podobě periodický. Nicméně existuje i speciální periodický sled Diracových impulsů. Více o FT pojednává například [4].

### 2.4.1 Diskrétní Fourierova transformace (DFT)

Diskétní Fourierova transformace představuje zjednodušení vlastností diskrétní Fourierovy řady zavedením konečné délky signálu. Periodizujeme vstupní signál

$$\overline{x}[n] = x[nmodN]$$

a koeficienty Fourierovy řady určíme pomocí:

$$\tilde{X}[k] = \sum_{n=0}^{N-1} x[n] * e^{-j\frac{2\pi}{N}kn}$$

$$x[k] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] * e^{j\frac{2\pi}{N}kn}$$

Tuto posloupnost ještě omezujeme okénkovou funkcí

$$X[k] = R_n \sum_{n=0}^{N-1} x[n] * e^{-j\frac{2\pi}{N}kn}$$

Proto když se dále bavíme o FTP, předpokládáme omezenou posloupnost i spektra.

# Kapitola 3

# Demonstrační příklady Fourierovy transformace

## 3.1 Porovnaní výpočtu koeficientů Fourierovy řady

V následující kapitole porovnám několik typů rovnic a jejich řešení pomocí semi-analytických výpočtů a Fourierových řad. Druhou část výpočtů budu provádět v programu CLTKSL s jeho grafickým rozhraním, které je implementováno v kapitole 4. Pro porovnání lze využít i původní verzi TKSL.

## 3.2 Přehled základních vzorců

Před začátkem řešení je dobré si připomenout některé základní vzorce pro převod goniometrických funkcí na Fourierovu řadu. Jsou uvedeny vzorce z [8] a [1].

$$\sin(3\alpha) = 3\sin(\alpha) - 4\sin^3(\alpha)$$
$$\cos(4\alpha) = 8\cos^4(\alpha) - 8\cos^2(\alpha) + 1$$
$$\sin(5\alpha) = 16\sin(\alpha)\cos(\alpha) - 12\sin(\alpha)\cos^2(\alpha) + \sin(\alpha)$$

$$sin(\alpha) + sin(\beta) = 2\sin(\frac{\alpha+\beta}{2})\cos(\frac{\alpha-\beta}{2})$$
$$cos(\alpha) - cos(\alpha) = -2\sin(\frac{\alpha+\beta}{2})\sin(\frac{\alpha-\beta}{2})$$

$$\sin(\alpha)\sin(\beta) = \frac{1}{2}[\cos(\alpha-\beta) - \cos(\alpha+\beta)]$$
$$\cos(\alpha)\sin(\beta) = \frac{1}{2}[\sin(\alpha+\beta) - \sin(\alpha+\beta)]$$

## 3.3 Příklad - mocnina goniometrické funkce

Jako první si zvolím příklad základní mocniny goniometrické funkce se zadáním $f(t) = 2\cos^4(t)$. Je evidentní, že $\omega = 1rad/s$.

### 3.3.1 Ruční výpočet

Při řešení budu aplikovat následující vzorec:

$$\cos^4(\alpha) = \frac{1}{8}[\cos(4\alpha) + 4\cos(2\alpha) + 3]$$

S jeho využitím dostanu výsledek:

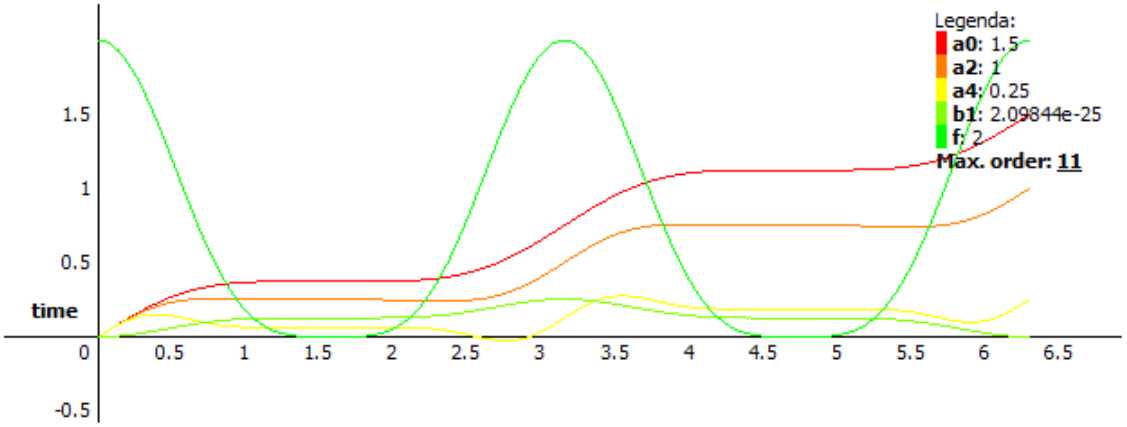
$$2\cos^4(t) = 4 * \frac{1}{8} * (\cos(4t) + 4\cos(2t) + 3) = \frac{1}{2}cos(4t) + 2cos(2t) + 3$$

Určili jsme koeficienty Fourierovy řady jako: $a_0 = 3, a_2 = 2, a_4 = \frac{1}{2}$, ostatní jsou nulové.

### 3.3.2 Výpočet pomocí nového uživatelského rozhraní CLTKSL

Pro výpočet musím zadat vstupní funkci ve tvaru: `f=2*cos(t)*cos(t)*cos(t)*cos(t);`

Dále využiji pomocné diferenciální rovnice (viz. kapitola 4.3). Pro řešení stačí nastavit krok 0.01 a maximální čas na hodnotu $2\pi$. Rozhraní vykreslí následující graf:




Obrázek 3.1: Příklad 1 - Výstup implementovaného grafického rozhraní

Dostal jsem očekávané výsledky, kdy pokud získané koeficienty $a_0, a_2$ a $a_4$ vynásobím hodnotou $f$, získám totožné výsledky jako u ručního výpočtu.

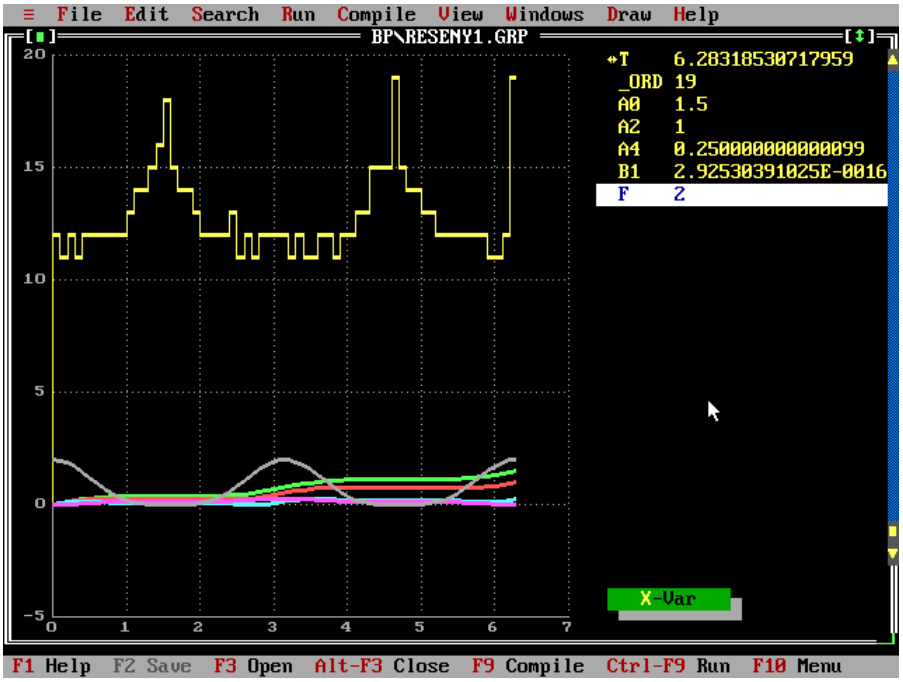
### 3.3.3 Ověření získaného grafu v TKSL

Na závěr pro potvrzení výsledků z nové verze TKSL/C ve spojení s grafickým rozhraním, provedu ověření výpočtu v původním TKSL/386 (testováno spuštěním pomocí programu DOSBox® na stejném počítači jako TKSL/C). Pro urychlení výpočtu jsem nastavil krok na 0.1, který by neměl být znatelný na výsledku.

Výsledek výpočtu je na obrázku 3.2, kde lze vidět, že získávám totožné výsledky jako v nově implementovaném rozhraní, čím je potvrzena důvěryhodnost grafu.

## 3.4 Příklad - součin goniometrických funkcí

Druhý demonstrační model řeší funkci $3\sin(2t)\sin(5t)\cos(4t)$, u kterého se na první pohled zdá analytické řešení, jako velmi složité. Jak ale bude uvedeno v následující podkapitole, jedná se o učebnicový příklad užití vzorce.




Obrázek 3.2: Příklad 1 - Výstup z TKSL

### 3.4.1 Ruční výpočet

Pro řešení využijeme vzorec:

$$\sin(\alpha)\sin(\beta)\cos(\gamma) = \frac{1}{4}[-\cos(\alpha+\beta-\gamma)+\cos(\beta+\gamma-\alpha)+\cos(\gamma+\alpha-\beta)-\cos(\alpha+\beta+\gamma)]$$

Kdy postupuji:

$$\begin{aligned} 3*\sin(2t)\sin(5t)\cos(4t) = \\ 3*\frac{1}{4}(-\cos(2t+5t-4t)+\cos(5t+4t-2t)+\cos(4t+2t-5t)-\cos(2t+5t+4t)) = \\ \frac{3}{4}(-\cos(3t)+\cos(7t)+\cos(t)-\cos(11t)) = \\ \frac{3}{4}\cos(t)-\frac{3}{4}\cos(3t)+\frac{3}{4}\cos(7t)-\frac{3}{4}\cos(11t) \end{aligned}$$

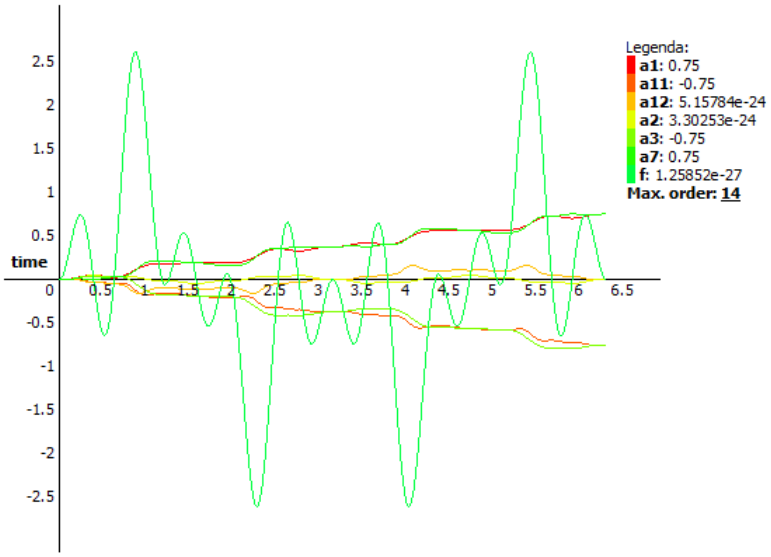
Koeficienty Fourierovy řady jsem určil $a_1 = \frac{3}{4}, a_3 = -\frac{3}{4}, a_7 = \frac{3}{4}$ a $a_{11} = -\frac{3}{4}$. Ostatní koeficienty jsou nulové.

### 3.4.2 Vyřešení v TKSL/C

Nyní budu demostrovat vyřešení zadané úlohy v TKSL/C a vytvořeném uživatelském rozhraní. Jako vstup musím uvést funkci ve tvaru: `f=3*sin(2*t)*sin(5*t)*cos(4*t);`

Krok opět ponecháme na hodnotě 0.01 a horní hranice času (stejně jako u předchozí úlohy) bude $2\pi$. Rovněž využiji diferenciální rovnice. Dostanu výstup na obrázku 3.3

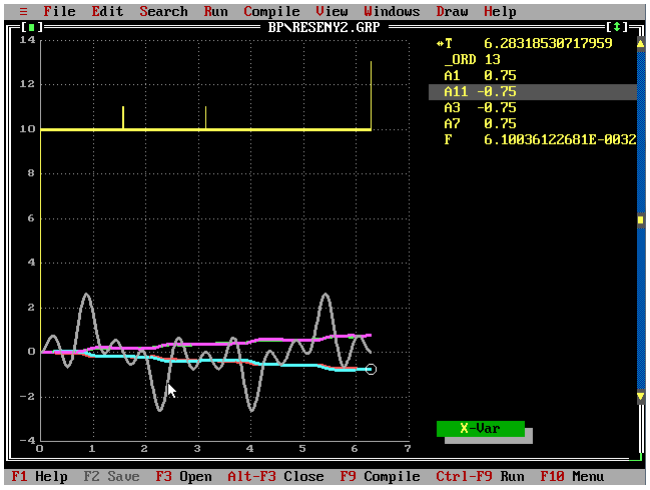
Opět byly správně určeny následující koeficienty: $a_1 = \frac{3}{4}, a_3 = -\frac{3}{4}, a_7 = \frac{3}{4}$ a $a_{11} = -\frac{3}{4}$.




Obrázek 3.3: Příklad 2 - Výstup implementovaného grafického rozhraní

### 3.4.3 Potvrzení výsledků v TKSL

I druhý příklad jsem prověřil pro porovnání v původní verzi TKSL. Aby bylo možné dosáhnout stejného výsledku, musel jsem nastavit krok na 0.01, kdy již výpočet na 2 jádrovém procesoru trval v řádku několika minut.




Obrázek 3.4: Příklad 2 - Výstup z TKSL

Ovšem i přes to jsem potvrdil, že získaný výpočet i zobrazený graf odpovídá naprogramovanému uživatelskému rozhraní pro CLTKSL.

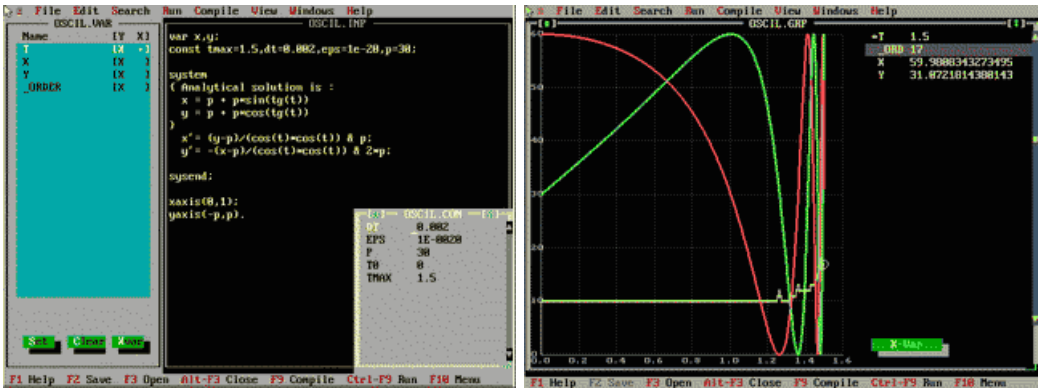
# Kapitola 4

# Program TKSL

## 4.1 Vývoj TKSL

Program TKSL představuje simulační systém pro testování algoritmů využívající Taylorovy řady k řešení diferenciálních rovnic a dalších problémů. Původně vytvořená verze nese označení TKSL/386 (dle platformy, na které byla vytvořena). V současné době se již vyvíjí její nástupce TKSL/C, někdy označován jako CLTKSL.

Prvotní TKSL/386 má vytvořené grafické uživatelské rozhraní v prostředí TurboVision. Dovoluje nastavovat řád metody, přesnost výpočtu a provádí detekci nespojitosti, kdy se během výpočtu dynamicky upravuje velikost integračního kroku. Na začátku je nutné si přeložit vstupní vzorový kód a následně má uživatel možnost spustit simulaci. Jejím výstupem je graf, kde lze odečítat hodnoty v bodech, které jsou význačné z hlediska grafu.




Obrázek 4.1: Náhledy programu TKSL (zdroj: http://www.fit.vutbr.cz/ kunovsky/TKSL/)

Tyto obrázky představují oficiální snímky obrazovky. TKSL/386 je naprogramován v jazyce Pascal, a proto je již v době dnešních výkonných počítačů nedostatečný.

Nyní probíhá vývoj TKSL/C (někdy označován jako CLTKSL), který je psán v programovacím jazyce C++ a tudíž lze finální řešení použít i mimo operační systémy Microsoft Windows. Zároveň se řeší i některé problémy původní verze. Mezi ně patří například:

- odstranění limitu počtu rovnic
- zjednodušená syntaxe vstupu

- možnost začlenění do různých programů a uživatelských rozhraní
- možnost použití víceslovní aritmetiky
- přenositelnost na různé platformy počítačů

V současné době neexistuje oficiální grafické uživatelské rozhraní a CLTKSL se ovládá z příkazové řádky. V rámci různých projektů, proto vznikají různé varianty GUI určené pro konkrétní řešení.

## 4.2 Parametry programu TKSL/C

Zde uváděná syntaxe platí pro aktuální verzi TKSL/C, která je součástí přiložených zdrojových kódů v binární podobě, spustitelná na platformě MS Windows 7.

```
cltksl.exe PARAMTERY vstup
```

Program vypisuje data na standardní výstup a je tudíž vhodné si je přesměrovávat do souboru, pro další výpočty, například:

```
cltksl.exe PARAMETRY vstup.tksl > vystup.txt
```

Význam jednotlivých parametrů:

| | | |
|---|---|---|
| -h | –help | vypíše nápovědu |
| -V | –version | informace o aktuální verzi programu |
| -s | –step==DOUBLE | nastavení délky kroku |
| -t | –time=STRING | nastavení koncového času |
| -W | –number–width=INT | nastavení délky čísel |
| -O | –max-order=DOUBLE | maximální řád metody |
| -A | –accuracy=DOUBLE | nastavení přesnosti výpočtu |
| -Z | –zeros=INT | počet nulových kroků, který detekuje konec kroku |
| -w | –text-width=INT | nastavení délky čísel |
| -p | –text-precision=INT | počet desetinných míst na výstupu |
| -f | –output-format=STRING | formát výstupu |
| -c | –convergency=INT | kontrola špatné konvergence |
| -i | –increase-step=DOUBLE | inkremenrace kroku pro malý řád |
| -x | –xml | výstup bude v XML formátu |
| vstup | | standartní vstup |
| vystup | | výstup programu |

## 4.3 TKSL a jeho řešení Fourierovy transformace

Program TKSL (stejně jako jeho nástupce TKSL/C) řeší problematiku Fourierových transformací pomocí převodu na diferenciální rovnice. Ty se vyhodnocují na základě numerické metody Taylorova rozvoje.

Vzhledem ke stále neexistujícímu grafickému rozhraní lze vstupní rovnice zadávat přímo do příkazové řádky nebo do externího souboru, který pak přesměrujeme jako vstup skriptu. Moje řešení (viz. kapitola 6) upřednostňuje druhou možnost - rovnice se ukládají do externího souboru, kde je uživatel může editovat. Po uložení jsou předány na vstup CLTKSL, které je vyhodnotí. Rovnice se zadávají ve formátu:

$$b1' = k * f * sin(om * t)\&0;$$

Jedná se tedy o již zmiňovanou diferenciální rovnici se zadanou počáteční podmínkou (b1(0) = 0). Pro usnadnění zadávání rovnic je možné zadávat i konstanty:

$$PI = 3.1415926535897932385;$$

Celý vstupní zdrojový soubor může mít následující podobu:

$/ * konstanty * /$
$om = 1;$

$/ * zadanafunkce * /$
$f = 0.33 * cos(t) + 2 * cos(2 * t) + 0.8 * sin(t) + 2.5 * sin(2 * t);$

$/ * pomocnefunkce * /$
$a0' = k * f\&0;$
$b1' = k * f * sin(om * t)\&0;$
$a1' = k * f * cos(om * t)\&0;$
$b2' = k * f * sin(2 * om * t)\&0;$
$a2' = k * f * cos(2 * om * t)\&0;$
$a3' = k * f * cos(3 * om * t)\&0;$
$b3' = k * f * sin(3 * om * t)\&0;$
$a4' = k * f * cos(4 * om * t)\&0;$
$b4' = k * f * sin(4 * om * t)\&0;$
$a5' = k * f * cos(5 * om * t)\&0;$
$b5' = k * f * sin(5 * om * t)\&0;$
$a6' = k * f * cos(6 * om * t)\&0;$

Vlastní zpracování pak provádí program zadáním příkazu:

$$cltksl.exe - t6.283185 - s0.001 - x - fa0 : b1 : a2input.tksl > output.xml$$

Ten provede simulaci výpočtů s následujícími parametry:

- časová osa bude od 0 po 2*$\pi$ (je vložena zaokrouhlená hodnota - platí, že čím větší počet desetinných míst člověk zadá, tím větší přesnost výpočtu dostane)
- derivační krok je nastaven na 0.001, což způsobí delší provádění výpočtu, ale kladně ovlivní jeho přesnost
- výstup bude ve formátu XML (příklad vzorového výstupu níže)
- požaduje vytisknout pouze hodnoty a0, b1 a a2
- vstupní data jsou v souboru `input.tksl`

- výstup bude přesměrován do souboru `output.xml`

Program vypočítá hodnoty celého Taylorova rozvoje, nicméně pro Fourierovu transformaci jsou zásadní až data z posledního kroku (v našem případě $2\pi$). Výstup ve formátu XML je dle následujícího vzorku:

$$
\begin{aligned}
&< tksl - outputversion =' 1.0' > \\
&< infophase =' parse' > C : /clTKSL/pr3.tksl < /info > \\
&< infophase =' parse' > C : /clTKSL/pr3.tksl - O.K. < /info > \\
&< infophase =' parse' > completed < /info > \\
&< results > \\
&< rid =' t'v =' 0.0000000000e + 00'/ > \\
&< rid =' a1'v =' 0.0000000000e + 00'/ > \\
&< rid =' b1'v =' 0.0000000000e + 00'/ > \\
&< rid =' f'v =' 0.0000000000e + 00'/ > \\
&< rid =' \#'v =' 0'/ > \\
&< /results > \\
&< /tksl - output >
\end{aligned}
$$

Položka results odpovídá jednomu řádku klasického výstupu a také jednomu derivačnímu kroku. Pro FT je pro nás zásadní až poslední krok, nicméně ostatní hodnoty jsou důležité pro vykreslení grafu.

# Kapitola 5

# Porovnání s existujícím programovým vybavením

Pro zpracování Fourierovy transformace neexistuje ve světovém měřítku známý specifický software. Velmi často se tak pro tuto problematiku využívájí programy Matlab, Maple a TKSL.

Transformaci lze vypočítat i ručně, neboli analyticky, kdy dosazujeme hodnoty do definičních vztahů. V tomto případě mám k dispozici pouze omezené množství funkcí, které lze integrovat. Větší rozsahy funkcí lze získat využitím výpočetní techniky v kombinaci s matematickým softwarem (například programy zmíněné v předchozím odstavci).

## 5.1 MATLAB

MATLAB patří mezi interaktivní programovací jazyky pro matematické, technické a vědecké výpočty, modelování a mnohé další využití. Byl vyvinut firmou MathWorks v 80-tých letech 20. století. Jeho zdrojové kódy představují sekvenci volání jednotlivých funkcí.


Obrázek 5.1: Logo společnosti MathWorks

Mezi široké výpočetní možnosti programu MATLAB patří i funkce fft pro Fourierovu transformaci. Ta provádí výpočet spektra, neboli koeficientů FŘ, v komplexní rovině. Funkce pracuje na základě algoritmu Fast Fourier Transformation (rychlá Fourierova transformace), který je definován pro diskrétní čas. Proto je nutné navzorkovat funkci tak, aby splňovala vzorkovací teorém. Vzhledem k tomu, že se převod provede symetricky, určuje počet vzorků polovinu koeficientů.

V následující kapitole bude ukázán názorný příklad FT. Aby bylo možno určit získání správných výsledků, je nutné použít funkce, u nichž je známá Fourierova řadu. Některé z těchto funkcí jsou uvedeny v tabulce v kapitole 2.3.2 převzaté z [2].

### 5.1.1 Vyřešení úlohy v MATLABu

Pro demonstraci řešení si zvolíme následující rovnici:

$$f(t) = \frac{1}{16} * (10 * \cos(t) + 5 * \cos(3t) + \cos(5t))$$

pro kterou mám za úkol vytvořit Fourierovu transformaci. Pro jednodušší určení koeficientů si zadanou rovnici upravím do tvaru:

$$f(t) = 0.625 * \cos(t) + 0.3125 * \cos(3t) + 0.0625 * \cos(5t)$$

kde jsou již zřejmé koeficienty Fourierovy řady. Vyjdu z definičního vztahu:

$$f(t) = \frac{a_0}{2} + a_1 * \cos(\omega t) + b_1 * sin(\omega t) + a_2 * cos(2\omega t) + b_2 * sin(2\omega t) + \ldots$$

kde dostanu výsledky koeficientů $a_0 = 0, a_1 = 0.625, a_3 = 0.3125, a_5 = 0.0625$. Ostatní jsou nulové.

### 5.1.2 Zdrojový kód příkladu

Nyní se zaměřím na část zdrojového kódu v jazyce MATLAB (celý kód je obsažen na přiloženém CD). Na začátku řešení je třeba si navzorkovat vstupní funkci pro získání konkrétních diskrétních hodnot. Velikost kroku si zvolím 0.01 a počet vzorků (N) 16.

```
T_MAX=1;
F=16;
T=1/F;
N=16;
% navzorkovani casu
t=0:0.01:T_MAX;
% diskretni honoty casu
dt=0:T:T*(N-1);
f=0.625*sin(pi*t)+0.3125*sin(3*pi*t)+0.0625*sin(5*pi*t);
```

Po navzorkování diskrétních hodnot vstupní funkce (uložené v proměnné dt) aplikuji tuto proměnnou na vstupní funkci f. Na takto upravený signál použiji funkci fft, kde dostanem výsledek výpočtu spektra funkce, který si uložím od proměnné S.

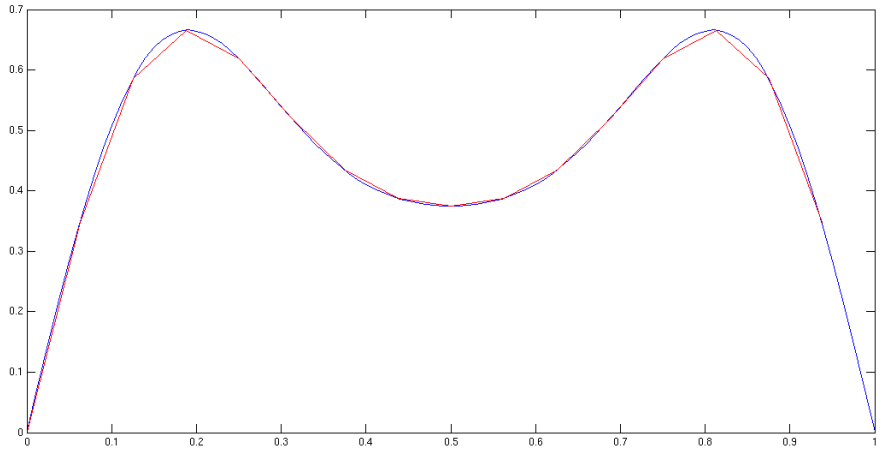
```
ns=0.625*sin(pi*dt)+0.3125*sin(3*pi*dt)+0.0625*sin(5*pi*dt);
S=fft(ns);
```

Výsledek mám v komplexní oblasti, proto je pomocí následujících vztahů převedu do reálné (do proměnné A budou uloženy koeficienty $a_n$, v proměnné B koeficienty $b_n$):

```
A=real(S)*(2/N);
B=imag(S)*(2/N);
```

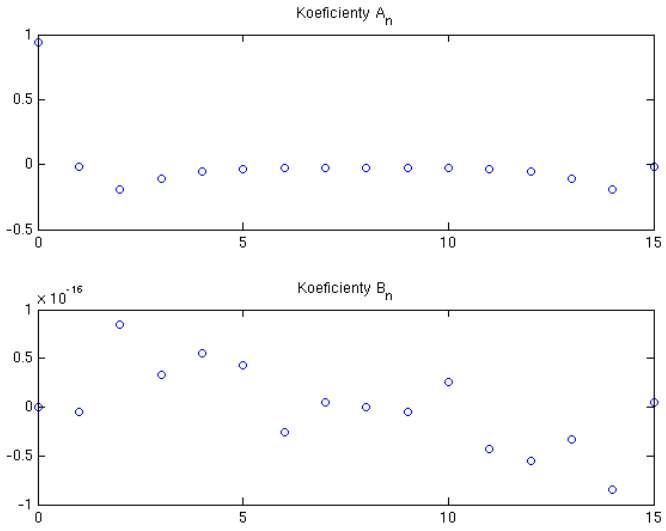
### 5.1.3 Zobrazení výsledků transformace

Ve zdrojovém kódu se jako první vykreslí zadaná funkce `f` (modrá křivka) a její diskrétní podoba `ns`. Výsledek z MATLABu verze 7.10.0 (běžícího na školním serveru merlin) je na obrázku 5.2.



Obrázek 5.2: Vstupní funkce f a její diskrétní podoba z navzorkovaného času

Data převedená z komplexní oblasti (vyjadřující koeficienty Fourierovy řady) jsou znázorněny na následujících grafech. Představují výpis hodnot proměnných A a B. Viz obrázek 5.3.




Obrázek 5.3: Hodnoty koeficientů $a_k$ a $b_k$

## 5.2 Maple

Maple je matematický program, který je určen především pro matematické operace, a proto nemá žádnou funkci, která by umožňovala výpočet Fourierovy řady. Vyvíjí jej Kanadská firma Maplesoft. Nicméně se jedná o mocný nástroj pro řešení mnoha problémů, včetně

výpočtů určitých i neurčitých integrálů. Díky tomu lze určit alespoň jednotlivé kroky Fourierovy transformace.


Obrázek 5.4: Logo společnosti Maplesoft

Při výpočtu je potřeba sestavit definiční rovnice, které už pak program vypočítá automaticky. Vzhledem k velké omezenosti rozsahu Maplu je nutné volit krok transformace tak, aby nebyl příliš široký. Na internetu jsou sice již zaznamenány pokusy o vytvoření knihovny díky které by šel výpočet Fourierových řad řešit nějakým obecnějším způsobem, nicméně v současné době není k dispozici žádné univerzálnější řešení. Zde uvedené řešení je proto pouze pro zadaný příklad, nelze jej použít jako obecný postup.

### 5.2.1 Vyřešení úlohy v Maplu

Pro řešení v Maplu je třeba si zvolit funkci, která bude splňovat základní požadavek - musí být sudá na zadaném intervalu, aby ji bylo možné v tomto programem zpracovat. Je tedy nutné zvolit typicky sudou funkci $2x^2$ na intervalu $-\pi$,$\pi$. Protože jsem si zvolil sudou funkci, tak můžu rovnou říci, že koeficienty Fourierovy řady $b_k$ jsou nulové. Správnost výpočtu pak můžeme ověřit ručně.

Ve výpočtu je potřebné použít knihovní funkce Maplu pro výpočet určitého integrálu `int`. Je nutné si také definovat počet koeficientů Fourierovy řady, který bude reprezentovat proměnná n. Následucíjím příkazem do ní přiřadíme celá čísla.

```
assume(n, integer);
```

### 5.2.2 Zdrojový kód úlohy v Maplu

Vytvořím si rovnice pro koeficienty za použití definic díky možnosti přiřazovat hodnoty přímo do proměnných. Musím rozdělit výpočet do 3 proměnných pro $a_0, a_n$ a $b_n$ (i když, jak předpokládám, tak $b_n$ bude nulová).

```
a0 := int(2*x^2, x = -Pi..Pi);
an := int(2*cos(n*x)*x^2, x=-Pi..Pi);
bn := int(2*sin(n*x)*x^2, x=-Pi..Pi);
```

Nyní lze použít funkci `evalf`, která vyhodnotí číselně výrazy.

```
a_0 := evalf(a0/Pi);
a_n := evalf(an/Pi);
b_n := evalf(bn/Pi);
```

Sečtu výsledky Fourierovy řady a sečtu koeficienty. Následně opět vytvořím definici obecné funkce, která bude dostávat jako parametr n (tj. počet koeficientů).

```
# secteni koeficientu
sum_koef := (1/2)*a_0;
```

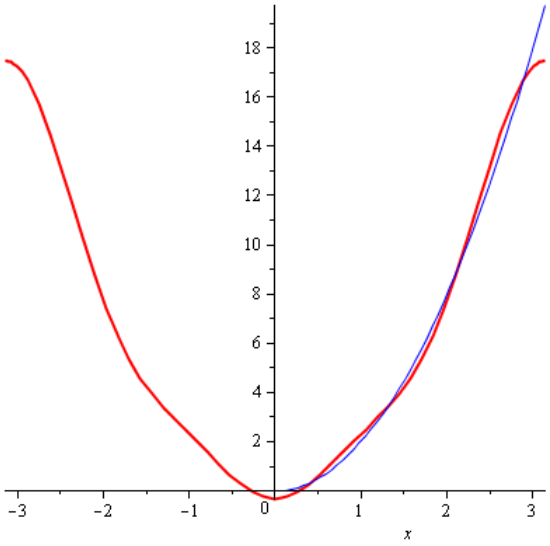
```
sum_koef := sum_koef+Sum(a_n*cos(n*x)+b_n*sin(n*x), n=1..infinity);

# vytvoreni funkce pro FR
four := m->a_0/2+Sum(a_n*cos(n*x)+b_n*sin(n*x), n=1..m);
```

### 5.2.3 Zobrazení výsledků

Vzhledem k výhodnému výstupnímu tvaru Maplu jsou výsledky koeficientů zobrazeny okamžitě. Dostávám následující hodnoty $a_0 = 13.15947254, a_N = \frac{-8}{n^2} ab_N = 0$.

Často je také výhodné nechat si vykreslit graf. Pomocí následujících příkazů je vykreslen graf 5.5, kde je vidět průběh původní funkce $(2 * x^2)$ zeleně a červenou barvou pak náhrada funkce příslušné řady koeficientů Fourierovy řady:

```
with(plots);
orig := plot(2*x^2, x=0..1*Pi, color=blue):
ft := plot(four(3), x=-Pi..Pi, color=red, thickness=2):
display(ft, orig);
```




Obrázek 5.5: Výstup příkladu řešené v programu Maple

### 5.2.4 Ověření správnosti výpočtu

Jak již bylo v kapitole 5.2.1 zmíněno, bude provedeno ověření výpočtu ručním výpočtem. Při následujících úpravách je využito integrování výrazů pomocí metody per partes (vzorce využity z [1]).

$$a_0 = \frac{1}{T} \int_{-T}^{T} f(t)dt$$

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} 2t^2 dt = \frac{1}{\pi} [2\frac{t^3}{3}]_{-\pi}^{\pi} = \frac{1}{\pi} * 41.3417 = 13.1595$$

$$a_n = \frac{1}{T}\int -T^T f(t) * \cos(nt)dt$$
$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} 2t^2 * \cos(nt)dt = \frac{2}{\pi}[-\frac{t^2}{n} * (\sin(nt)) - \frac{2}{n^3} * (\sin(nt)) + \frac{2t}{n^2} * (cos(nt))]_{-\pi}^{\pi} =$$
$$\frac{2}{\pi}[\frac{2\pi}{n^2} * (\cos(\pi n) - \frac{2 * (-\pi)}{n^2} * (\cos(-\pi n)))]_{-\pi}^{\pi} = \frac{2}{\pi} * \frac{4\pi}{n^2} = \frac{8}{n^2}$$

$$b_n = \frac{1}{T}\int -T^T f(t) * \sin(nt)dt$$
$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} 2t^2 * \sin(nt)dt = \frac{2}{\pi}[-\frac{t^2}{n} * (\cos(nt)) + \frac{2}{n^3} * (\cos(nt)) - \frac{2t}{n^2} * (\sin(nt))]_{-\pi}^{\pi} =$$
$$\frac{2}{\pi}(-\frac{\pi^2}{n} + \frac{2}{n^3} + \frac{\pi^2}{n} - \frac{2}{n^3}) = \frac{2}{\pi} * 0 = 0$$

## 5.3 TKSL a TKSL/C

Program TKSL je aplikace vyvíjená pro řešení systémů diferenciálních rovnic. Jejím základem je Taylorův rozvor (o něm pojedává [6]). TKSL/C je novější verze TKSL přepsaná do jazyka C kvůli odstranění některých nedostatků starší verze a přenositelnosti aplikace mezi operačními systémy. Více o vývoji aplikace je popsáno v kapitole 4.

U tohoto programu vytváří hlavní omezení nutnost převést soustavu rovnic pro výpočet koeficientů na soustavu diferenciálních rovnic. Rovnice pro Fourierovu řadu mají tvar:

$$a_0 = \frac{1}{T}\int_0^{2T} f(t)dt$$
$$a_n = \frac{1}{T}\int_0^{2T} f(t)\cos(n\omega t)dt$$
$$b_n = \frac{1}{T}\int_0^{2T} f(t)\sin(n\omega t)dt$$

kde $n \in Zan > 0$. O této problematice se pojednává v kapitole 2.2. Aby bylo možné se vyhnout vyčíslování určitého integrálu, derivujeme obě strany rovnice, kdy všechny předchozí rovnice lze přesat do soustavy diferenciálních rovnic:

$$a_0' = \frac{1}{T} * f(t)$$
$$a_n' = \frac{1}{T} * f(t) * cos(n\omega t)$$
$$b_n' = \frac{1}{T} * f(t) * sin(n\omega t)$$

U těchto rovnic jsou počátky na nulové hodnotě, $\omega = \frac{2\pi}{T}$. V těchto případech přechází integrační interval do horní meze ($T_{max} = 2T$) [5].

### 5.3.1 Vyřešení úlohy v TKSL

Pro vyřešení úlohy v TKSL využiji poslední dostupnou verzi CLTKSL a mého uživatelského rozhraní pro vyobrazení výsledků. Zadání zvolím stejné jako u MATLABu, čili:

$$f(t) = \frac{1}{16} * (10 * \cos(t) + 5 * \cos(3t) + cos(5t))$$

Výsledky pak budu moci porovnat s výstupy z MATLABu. Pro zjednodušení mě ovšem bude zajímat pouze prvních 5 členů Fourierovy řady. Zdrojový kód bohužel není zpětně kompatibilní pro starší TKSL, nicméně novou verzi lze již považovat za dostatečně stabilní a spolehlivou.

Do zdrojového kódu vložím zadanou funkci ve tvaru:

```
f=1/16*(10*cos(t)+5*cos(3*t)+cos(5*t));
```

A pak ještě vzorce pro diferenciální rovnice pro výpočet koeficientů:

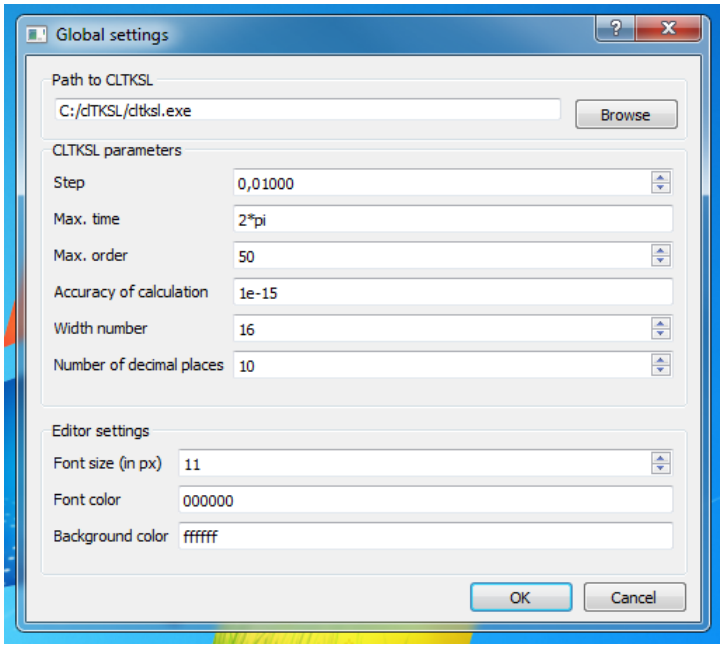
```
/* kontanty pro reseni */
k = 1;
om = 1;

a0'=k*f & 0;
a1'=k*f*cos(om*t) & 0;
a2'=k*f*cos(2*om*t) & 0;
a3'=k*f*cos(3*om*t) & 0;
a4'=k*f*cos(4*om*t) & 0;
a5'=k*f*cos(5*om*t) & 0;
b1'=k*f*sin(om*t) & 0;
b2'=k*f*sin(2*om*t) & 0;
b3'=k*f*sin(3*om*t) & 0;
b4'=k*f*sin(4*om*t) & 0;
b5'=k*f*sin(5*om*t) & 0;
```

Po spuštění rozhraní si otevřu soubor se zdrojovým kódem, který je součástí přiloženého CD. V menu Settings - Global settings nastavím správně cestu k binární TKSL/C aplikaci, dále pak krok (zvolili jsme s=0.01) a čas ($2\pi$). Ukázka nastavení je na obrázku 5.6.

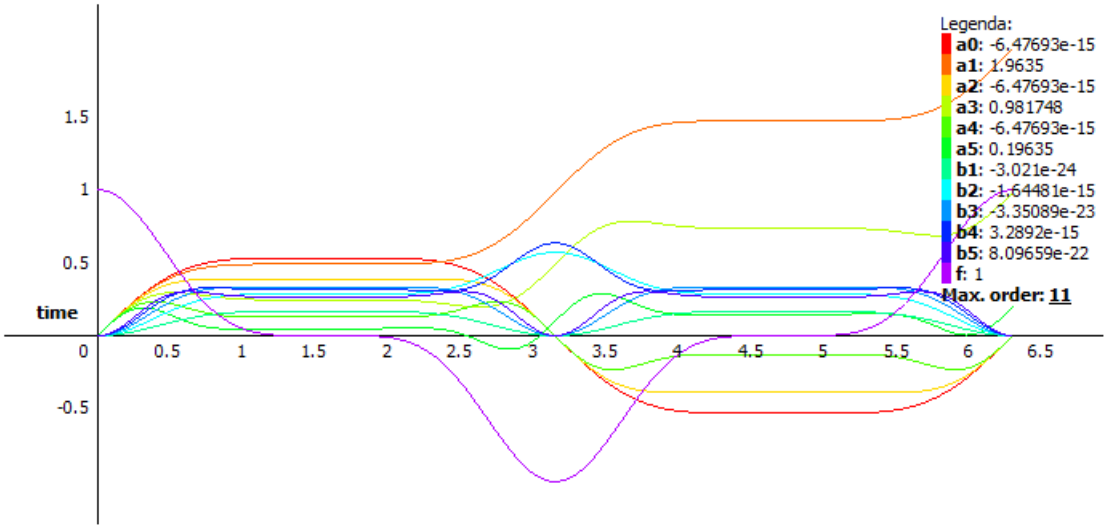
### 5.3.2 Zobrazení výsledku

Jak bylo v předchozí kapitole zmíněno, pro zobrazení jsem využil mnou vytvořené rozhraní, které vykreslí graf a vypíše hodnoty v posledním kroku derivace (tj. hodnoty Fourierovy transformace). Viz. obrázek 5.7.

Výstup lze uložit jako PNG obrázek přímo z aplikace pomocí nabídky File - Save graph as image. Prozatím u TKSL/C není možné nastavit počáteční čas, proto by například selhal příklad z programu Maple (5.2.1), který se řeší na intervalu $-2\pi, 2\pi$. Nicméně poslední verze už umožňuje zadat záporný čas a kroky se tak budou odečítat.




Obrázek 5.6: Ukázka nastavení rozhraní pro tento příklad




Obrázek 5.7: Výstup rozhraní pro CLTKSL

# Kapitola 6

# Uživatelské rozhraní

Jak bylo v textu práce již několikrát zmíněno, nové verze TKSL/C stále nemá existující grafické rozhraní. Zároveň je nutno podotknout, že i většina již existujícího softwaru, také nemá příliš uživatelsky příjemné rozhraní pro výpočty Fourierovy transformace. Proto jeden z bodů této bakalářské práce byl vytvoření grafického uživatelského rozhraní pro program CLTKSL. Po úvodní analýze a poradě s autory programu jsem dospěl k názoru, že GUI lze naprogramovat samostatně od aplikace TKSL/C.

V původním programu byla pouze provedena úprava výstupu, která nově umožňuje výstup ve formátu XML, který se snadněji parsuje pro další zpracování v grafickém rozhraní.

## 6.1 Definování požadavků na rozhraní

Po několika poradách s vedoucím práce a autory programu TKSL jsem si definoval několik základních požadavků pro mnou vytvořené rozhraní:

- spustitelnost pod MS Windows s podporou i některé z UNIXových distribucí
- rozhraní bude pracovat s panely souborů, tj. je možné mít otevřeno více souborů v jedné instanci programu
- soubory lze vytvářet, editovat, ukládat změny
- uživatel bude moci nastavovat parametry CLTKSL (délka kroku, maximální čas)
- aplikace vykreslí ze zpracovaných dat graf
- lze vybrat funkce, které budou zobrazeny v grafu
- systém umožňuje zvětšovat (přibližovat) i naopak oddalovat graf (zoomování)
- výsledný graf lze uložit jako obrázek do souboru

## 6.2 Návrh aplikace

Po stanovení požadavků na implementaci bylo nutné vytvořit návrh aplikace. V souvislosti s tím bylo nutné si zvolit i programovací prostředí, ve kterém bude rozhraní vyvíjeno. Informace o syntaxi C++ jsem čerpal z [9].

Vzhledem k požadavkům jsem se rozhodnul využít jeden projekt z povinně volitelného předmětu Seminář C++, kdy bylo naším úkolem vytvořit projekt tabulkového editoru, který měl také pracovat s více otevřenými soubory současně. Od tohoto jsem vytvořil návrh na 2 oddělené hlavní třídy programu (pro vlastní okno aplikace a jednotlivé soubory).

Další bodem bylo určení, jak se bude volat externí TKSL/C s jeho parametry. Protože jsem se rozhodl využívat volání funkce `popen`, rozhodnul jsem se, že vstupní data se budou posílat uložená v soboru jako jeden z parametrů skriptu. Od toho plyne kontrola, aby uživatel nemohl v rozhraní provést změny v souboru, které by neuložil a pokoušel by se skript spustit.

## 6.3 Volba programovacího prostředí

Vzhled k požadavku na multiplatformost jsem se rozhodl rozhraní implementovat v prostředí knihovny Qt. Jedná se o jednu z nejznámějších knihoven pro vytváření grafických uživatelských rozhraní. Informace z této kapitoly jsem čerpal z [7].


Obrázek 6.1: Logo Qt

Původní toolkit vytvořila firma Trolltech, která jej prodala firmě Nokia. Qt je použitelný na různých platformách. Jedná se knihovnu C++ a zvolil jsem si ji z několika důvodů. Umožňuje zpracovávat XML, má přehledné prostředí pro tvorbu rozhraní a spoustu tříd, které umožňují i některé netriviální operace. Velkým kladem je samozřejmě i přehledná nápověda dostupná na webu http://qt-project.org/doc/qt-4.8/

## 6.4 Implementace

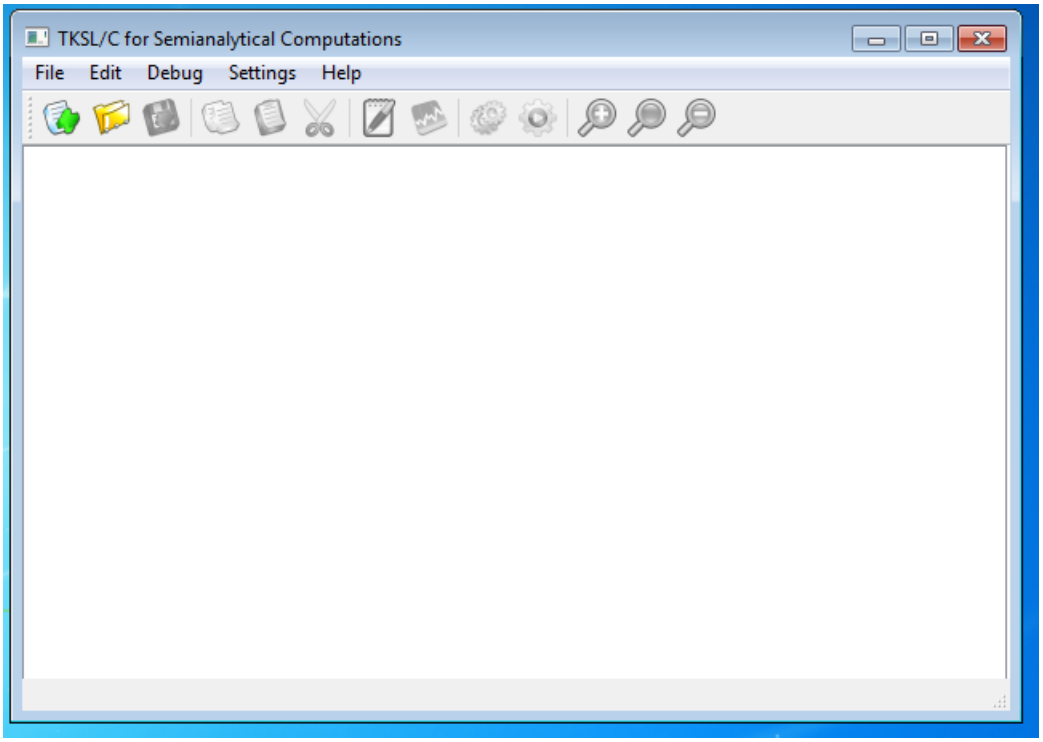
V následujících kapitolách bude popsána implementace jednotlivých částí rozhraní.

### 6.4.1 Hlavní aplikace

Hlavní okno aplikace, které naběhne po startu aplikace je logicky členěno do 4 horizontálních oddílů - horní hlavní menu, pod ním ikonové menu, následuje hlavní část, kde se budou zobrazovat panely souborů, jejich zdrojové texty a grafy. Jako poslední je dolní lišta, která je využita například pro nápovědy v menu a vypisování souřadnic v grafu.

Popisky jsou v anglickém jazyce, které odpovídají textům z původního rozhraní programu TKSL/386.

V obrázku 6.2 je zřejmé, že aplikace využívá předdefinované třídy knihovny Qt. Jedná se o tyto:




Obrázek 6.2: Hlavní okno aplikace po startu

- QMainWindow - hlavní okno aplikace
- QMenuBar - horní hlavní vyjížděcí menu, na které jsou navázané sloty akcí
- QToolBar - ikonové menu
- QStatusBar - spodní lišta
- QWidget - tento widget je použit pro hlavní část okna a dále obsahuje následující třídy
- QTabWidget - obsahuje jednotlivé taby se soubory
- QStackedWidget - součástí každého tabu, zapouzdřuje v sobě zdrojový text a graf
- QTextEdit - slouží pro editaci zdrojových textů (pro TKSL)
- QGraphicView - vykreslování grafu
- QString, QFile, QVector, QColor - práce s řetězci, soubory, vektory a barvami

Z programové části tvoří toto okno 2 třídy `MainWindow` a `tab`. První jmenovaná má následující strukturu:

```
public:
Ui::MainWindow *ui;
QVector<tab *> tab_vector; explicit MainWindow(QWidget *parent = 0); // vytvori
nove okno
```

```
MainWindow();
/// promenne pro globalni nastaveni
QString tksl_path;
float par_s;
QString par_t, par_a;
int par_o, par_w, par_p, font_size;
QString font_color, background;

void remove_first_tab();
int createTab(QString path);
void openFile(QString path, int index_tab);
int getActiveTab();
QString getFileFromPath(QString path);
int saveToFile(QString path, int index_tab);
void processTksl(bool compile);
virtual void loadSettings();
private slots:
void on_actionExit_triggered();
void on_actionOpen_triggered();
void on_actionNew_triggered();
void on_actionSave_triggered();
void on_actionSave_as_triggered();
void on_actionCompil_triggered();
void on_actionRun_triggered();
void on_actionSource_text_triggered();
void on_actionGraph_triggered();
void on_actionGlobal_triggered();
void on_actionAbout_triggered();
void on_actionAbout_Qt_triggered();
void on_actionAbout_GUI_triggered();
void on_actionAbout_TKSL_triggered();
void on_actionGraph_setting_triggered();
public slots:
void closeTab(int index);
void changeSourceText();
void myCopy();
void myCut();
void myPaste();
void changeTab(int index);
protected:
void resizeEvent(QResizeEvent *event);
void closeEvent(QCloseEvent *event);
```

Po spuštění programu se zavolá metoda remove_first_tab, která má za úkol odstranit první tab, který je vložen z původního návrhu rozhraní. Tím pádem program začne běžet vždy bez jakéhokoliv otevřeného souboru.

Pro běh aplikace je důležitá veřejná proměnná tab_vector. Ta reprezentuje pole objektů třídy tab (viz. dále), které představuje jednotlivé taby s otevřenými soubory.

Otevření souboru obstarává metoda openFile, kterou volá slot on_actionOpen_triggered(). Předá mu adresu souboru, který se má otevřít a také index tabu. Po prvním otevření souboru se aktivují akce kopírování, vkládání a vyjmutí textu do schránky. Zobrazený zdrojový text je samozřejmě možné editovat a ukládat provedené změny.

Kompilování a vykreslení grafu provádí společně metoda `processTksl`. Liší se pouze v místě ukončení, toto je jediné řešení, vzhledem k tomu, že program TKSL/C vypisuje na výstup data až po celém zpracování. Kompilování pouze zkontroluje, zda li je na výstupu potvrzení o úspěšném zpracování. Zatímco sestavení zároveň zavolá vykreslení grafu z aktuálních hodnot.

### 6.4.2 Zpracování výstupu z TKSL/C

Mnou navržené uživatelské rozhraní zpracovává data ve výstupním XML formátu. Prvotní kontrola pouze provede kontrolu, zda-li se na výstupu objeví řádky s informací o úspěšném zpracování zadaného vstupu (v našem případě vstupního souboru):

```
<tksl-output version='1.0'>
<info phase='parse'>C:/clTKSL/pr3.tksl</info>
<info phase='parse'>C/clTKSL/pr3.tksl - O.K.</info>
<info phase='parse'>completed</info>
```

Díky užití Qt toolkitu má rozhraní pro zpracování XML souborů třídu QXmlStreamReader. Ta na základě dodaných dat (v našem případě výstup z CLTKSL) rozloží XML do jednotlivých tokenů a ty lze pak pomocí cyklu jednoduše procházet. QXmlStreamReader zároveň kontroluje syntaxi XML, programátor se pouze stará o data, které si potřebuje vyparsovat.

V mém případě se vždy získávají data o jednotlivých hodnotách funkce v jednotlivém čase. Jedna položka `results` odpovídá vždy hodnotám v jednom čase:

```
<results>
<r id='t' v=' 0.0000000000e+00'/>
<r id='a1' v=' 0.0000000000e+00'/>
<r id='b1' v=' 0.0000000000e+00'/>
<r id='f' v=' 0.0000000000e+00'/>
<r id='#' v='0'/>
</results>
```

Tyto údaje se ukládají do datové struktury `QList<QMap <QString, float> >` data, která představuje asociativní pole. Během zpracování se zároveň určuje maximální a minimální hodnota dat na obou osách, na základě těchto hodnot se pak vykreslí do grafu osy.

### 6.4.3 Vykreslení grafu

Aplikace začne vykreslovat graf až poté co úspěšně zpracuje data z TKSL/C, tím odpadá problematika možných chyb (chybějící hodnoty, chyba ve výstupu aplikace CLTKSL). Graf se vykresluje do instance třídy `QGraphicsSceneGps` (jedná se třídu, která dědí drtinou většinu metod a proměnných od `QGraphicsScene`, kdy je navíc upraveno vypisování souřadnic myši na grafu). Osy se vykreslí pomocí metody `addLine(x1, y1, x2, y2)`, zatímco křivky jednotlivých funkcí vykresluje funkce addPath. Ta má výhodu, že ji netvoří pouze 2 body, zatímco vektor jednotlivých bodů.

Každá křivka má vždy jinou barvu. Tohoto efektu (důležitého pro rozlišení) je dosaženo pomocí vygenerování barev nasycením, využívá se metoda `setHsv` třídy `QColor`. Stejné barvy jsou i u popisku grafu.

Při každé změně velikosti okna (maximalizací apod.) je třeba graf překreslit, toto umožňuje metoda `scale(x, y)`, která scaluje v obou rozměrech.

U každého vykresleného grafu se v levém horním rohu vykresluje legenda grafu, která přiřazuje barvám křivek jména. Zároveň je zde uvedena hodnota posledního kroku, která je zásadní pro Fourierovu transformaci. Rovněž jsou do grafu pro popis uvedeny hodnoty na obou osách.

Aby bylo možné určit alespoň přibližnou hodnotu funkce v určitém bodě, je implementována následující věc. Při pohybu myši po grafu se do spodního stavového řádku vypisují právě tyto hodnoty. Nicméně není možné přesně určovat hodnoty z načtených dat, protože se jedná pouze o polohu myši v již vypsaném grafu.

### 6.4.4 Třída tab

Jak již bylo v kapitole 6.4.1 zmíněno, na zobrazování hlavního okna aplikace se podílí dvě třídy. O první `QMainWindow` bylo pojednáváno doposud. Instance třídy `tab` jsou uloženy ve vektoru `tab_vector`, který tvoří globální proměnnou třídy MainWindow.

Struktura třídy je následující:

```
class tab
{
public:
QWidget *current_tab;
QStackedWidget *stackedWidget;
QWidget *pageSourceText; // stranka se zdrojovym textem
QTextEdit *textEdit; // textovy editor
QLabel *labelSourceText;
QWidget *pageGraph;
QGraphicsView *graphicsView;
QGraphicsScene *scene; // scena pro graf
QLabel *labelGraph;
QLabel *legenda;
QList<QMap <QString, float> > data;
QMap<QString, bool> hide;
float t_min, t_max, v_min, v_max;
float zoom;
QLabel *time;
QVector<QLabel *> osa_x;
QVector<QLabel *> osa_y;
QString file_path;
bool isChange;
tab(QTabWidget *parent, QString path, Ui::MainWindow *ui); // kostruktor
tab(); // destruktor
void inputSourceText(QString &text); // vloz text do SourceText
int isActive(); // vraci index aktivniho tabu
void setActivity(int index); // nastavi aktivitu na zadany tab
void clearTab(); // vyprazdni tab
```

```
void drawGraph();
void resizeGraph(float zoom = 1.0);
};
```

Třída v sobě obsahuje jednotlivé objekty tabu - textovou oblast pro zdrojový kód (QTextEdit), objekty pro graf a popisky. Pro běh programu jsou důležité 2 metody `drawGraph()` a `resizeGraph(int)`. První vykresluje graf a druhá změní velikost grafu. Popis vykreslení grafu je již v kapitole 6.4.3.

### 6.4.5 Změna velikosti grafu

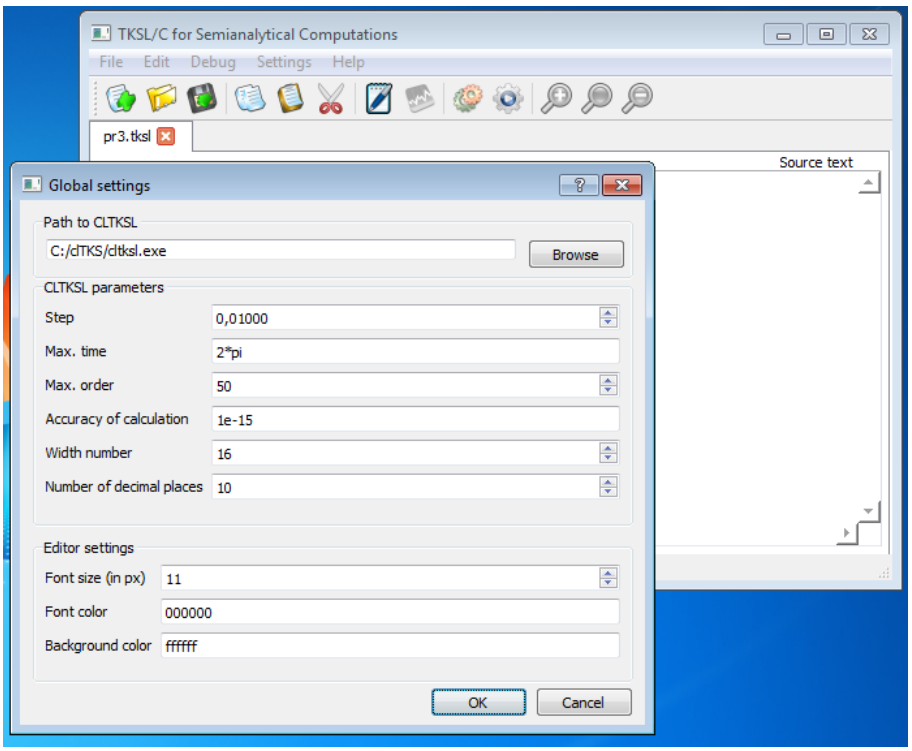
Při každé změně velikosti okna je třeba vždy celý graf překreslit. Změní se velikost QGraphicsView, podle nadřazeného prvku. Zároveň překreslení probíhá i při zoomování grafu.

Nejprve se určí, která osa se zvětšila/zmenšila více a podle ní se upraví velikost celého grafu. Toto je důležité, aby nedocházelo k deformaci grafu.

Po vlastní změně velikosti grafu je ještě třeba přesunout popisky v grafu (jako legendu, popis os, hodnoty) na správné místo. Některé statické texty (např. time) se pouze přesune. Ale hodnoty os se odstraní a vypočítává se nové umístění těchto labelů.

### 6.4.6 Nastavení externího TKSL/C

Aby bylo možné nějakým způsobem vyhovět požadavku na možné multiplatformní nasazení aplikace, je nutné vyřešit i cestu k binárnímu souboru CLTKSL. Navržené rozhraní toto řeší v globálním nastavení.



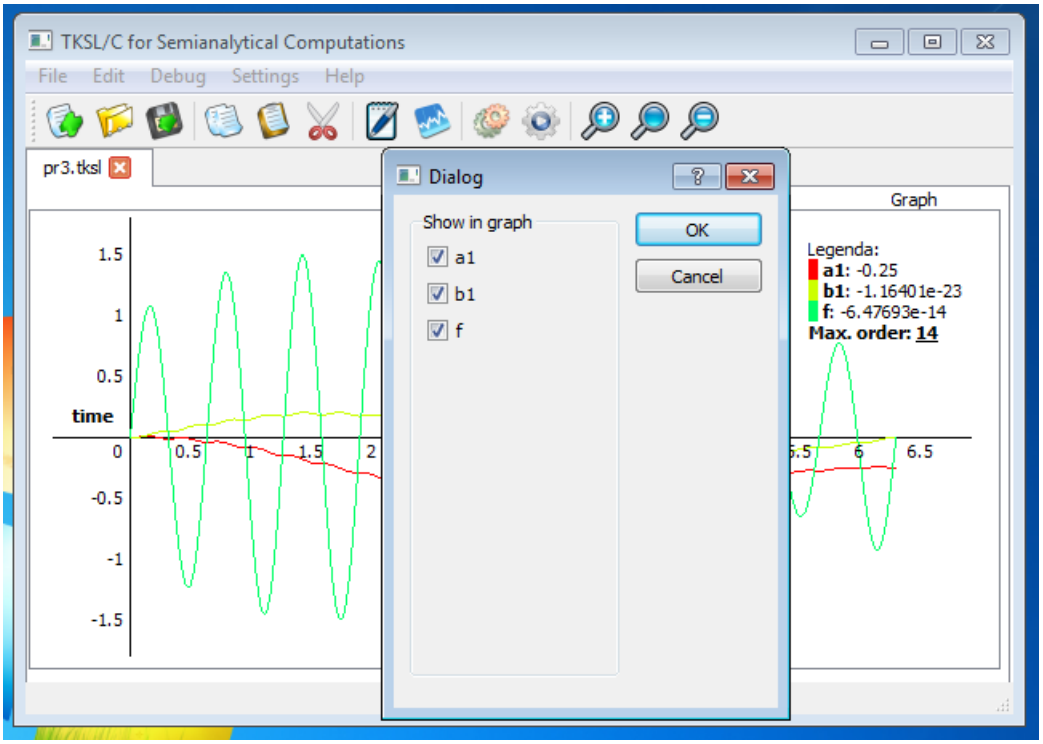

Obrázek 6.3: Příklad nastavení uživatelského rozhraní

Uživatel zadává celou cestu k CLTKSL, tím odpadá problém s různými souborovými systémy i různými názvy binárního souboru. Zároveň je zde možné nastavit parametry TKSL/C jako délku derivačního kroku, horní hranici času, přesnost výpočtu. Popis parametrů je v kapitole 4.2 (Parametry TKSL/C). Pro usnadnění zadání horní hranice času, je implementovaná podpora konstant PI, 2PI, PI/2 a PI/4, které pracují s Ludolfovým číslem s 50 desetinnými místy.

Spodní část slouží k nastavení editoru zdrojového kódu, které mohou uživateli zpříjemnit jeho práci s rozhraním, u kterého lze změnit barvu písma, velikost písma a barvu pozadí.

### 6.4.7 Nastavení grafu

Po prvním zkompilování zdrojového kódu a vykreslení grafu se uživateli zaktivní možnost editovat si nastavení grafu.




Obrázek 6.4: Nastavení jednotlivých grafů

Zde si uživatel může zvolit, které funkce si chce nechat vykreslit do grafu. Toto byl jeden z požadavků na rozhraní. Jistě to pomůže k zvýšení přehlednosti.

# Kapitola 7

# Hodnocení rozhraní

## 7.1 Dotazník

Pro objektivní vyhodnocení splnění požadavků na uživatelské rozhraní definovaných v kapitole 6.1 je třeba získat podněty a informace od uživatelů. U rozhraní nastával problém s testováním, jelikož by si uživatelé museli stáhnout a nainstalovat CLTKSL a poté spustit rozhraní. Proto jsem zvolil metodu, že testování probíhalo u mě na notebooku. Vytipoval jsem si několik odlišných skupin uživatelů, které lze roztřídit do následujících skupin:

- student FIT seznámený s problematikou Fouerierovy transformace
- student FIT, který FT blíže nezkoumal
- student vysoké školy z oblasti mimo IT
- ostatní

Uživatelé dostali spuštěnou aplikaci, kterou si mohli projít. Po zhruba 15 minutách dostali krátké zaškolení a jednoduchou demonstrační úlohu. Následovalo vyplnění dotazníku, jehož cílem bylo vyhodnotit splnění požadavků na rozhraní a jeho uživatelskou přívětivost. Ta je důležitá pro další možný vývoj aplikace. Dotazník obsahoval následující otázky:

1. Do které skupiny se řadíte? (student FIT se zájmem o FT, student FIT, student jiné VŠ, ostatní)
2. Jak hodnotíte své počítačové dovestnosti? (1-5, jako ve škole)
3. Jak hodnotíte přívětivost grafického uživatelského rozhraní? (1-5)
4. Jak hodnotíte intuitivnost rozhraní? (1-5)
5. Jak hodnotíte možnost práce s více otevřenými soubory? (1-5)
6. Jak reálně vidíte možnost využití rozhraní při výuce? (1-5)
7. Jak Vám vyhovuje možnost nastavení zobrazení grafu? (1-5)
8. Jak hodnotíte důvěru v zobrazené informace? (1-5)
9. Můžete vyjmenovat jednu věc, která Vás na rozhraní zaujala?
10. Můžete vyjmenovat jednu věc, která se Vám nelíbí?
11. Nějaké další poznatky k rozhraní?

## 7.2 Vyhodnocení

Do uzavření vyhodnocení bylo vyplněno 23 dotazníků. Průměrné známky z otázek 2-8 jsou shrnuté v následující tabulce (vyloučil jsem případy, kdy známka nebyla z množiny $\{1, 2, 3, 4, 5\}$). Mezi hodnotícími byli **2** studenti FIT se zájmem o FT, **8** studentů FIT, **6** studentů jiné VŠ/VOŠ a **7** ostatních.

| Číslo otázky | Otázka | Známka |
|---:|---|:---:|
| 2 | Počítačové dovednosti | 1,6087 |
| 3 | Přívětivost rozhraní | 1,6522 |
| 4 | Intuitivnost rozhraní | 1,9565 |
| 5 | Panel se soubory | 1,8696 |
| 6 | Využití ve výuce | 1,8261 |
| 7 | Nastavení grafu | 2,3478 |
| 8 | Důvěra v rozhraní | 1,5217 |

V odpovědích na otázky 9-11 byli spíše drobné detaily, které měly spíše charakter poznámek. Výraznou nespokojenost projevili 3 lidé. První si stěžoval na nemožnost uložení grafu do souboru (v době jeho testování tato funkce ještě nebyla naimplementována) a další poukázal na chybu při otevření více souborů.

Po vyhodnocení dotazníků lze tvrdit, že rozhraní splňuje vytyčené požadavky. Jedná se samozřejmě o věc názoru, ale dle známek je zřejmé, že rozhraní je uživatelsky přívětivé a dostatečně intuitivní.

# Kapitola 8

# Závěr

V průběhu vytváření práce jsem blíže nastudoval teorii Fourierovy řady a transformace. Rovněž jsem se seznámil s již existujícím softwarovým vybavením pro výpočty Fourierovy transformace. Tyto programy jsem se pokusil stručně popsat a v každém vyřešit jednu demonstrační úlohu. Zdrojové kódy příkladů jsou umístěny na přiloženém CD.

Velký důraz jsem kladl na program TKSL, včetně jeho novější varianty TKSL/C. Pro ni jsem vytvořil grafické uživatelské rozhraní (GUI), které se na základě kladného vyhodnocení dotazníku mezi uživateli podařilo implementovat. Rozhraní *TKSL/C for Semianalytical Computations* bylo otestováno na systémech Microsoft Windows Xp, Vista a 7 a na unixové distribuci Ubuntu 10.04, čímž se potvrdil požadavek na multiplatformovost. Práce splňuje všechny požadavky zadání. Vytvořené rozhraní lze využít i při výuce na naší fakukltě (např. v předmětu ITO - Teorier obvodů).

Během vytváření aplikace jsem si procvičil jazyk C++ a toolkit Qt. Zároveň také práci s principy objektového programování, které byli součástí kurzů Seminář C++ a Principy programovacích jazyků a OOP.

Jako jedno z možných rozšíření do budoucna by bylo možné optimalizovat čtení hodnot z grafu, kde se by zobrazovaly pouze hodnoty z křivek. Dále pak možný výpis hodnot z TKSL/C do tabulky pro jejich další možné studium.

# Literatura


[1] Bartsch, H.-J.: Matematické vzorce. 2006, iSBN 80-200-1448-9.

[2] Čížek, V.: Diskrétní Fourierova transformace a její použití. 1981, iSBN 0-401-98-1.

[3] Kopka, H.: *LaTeX: Podrobný průvodce*. Computer Press, Brno, 2004, iSBN 80-722-6973-9.

[4] Kunovský, J.: Modern Taylor Series Method. 1994, habilitační práce.

[5] Kureš, M.: Matematická analýza 1 [online]. `http://mathonline.fme.vutbr.cz/download.aspx?id_file=1050`, 2001-08-21 [cit. 2012-04-28].

[6] Musilová, J.; Musilová, P.: Matematika I pro porozumění i praxi. 2009, iSBN 978-80-214-3631-2.

[7] Nokia: Qt: Developer Network [online]. `http://http://qt-project.org/doc/`, 2012 [cit. 2012-04-28].

[8] Polák, J.: Přehled středoškolské matematiky. 2005, iSBN 80-7169-267-8.

[9] Prata, S.: Mistrovství v C++. 2007, iSBN 978-80-251-1749-1.

[10] Rábová, Z.; Hanáček, P.; Peringer, P.; aj.: Užitečné rady pro psaní odborného textu [online]. `http://www.fit.vutbr.cz/info/statnice/psani_textu.html`, 2008-12-01 [cit. 2008-04-28].

# Příloha A

# Obsah CD

Struktura přiloženého CD:

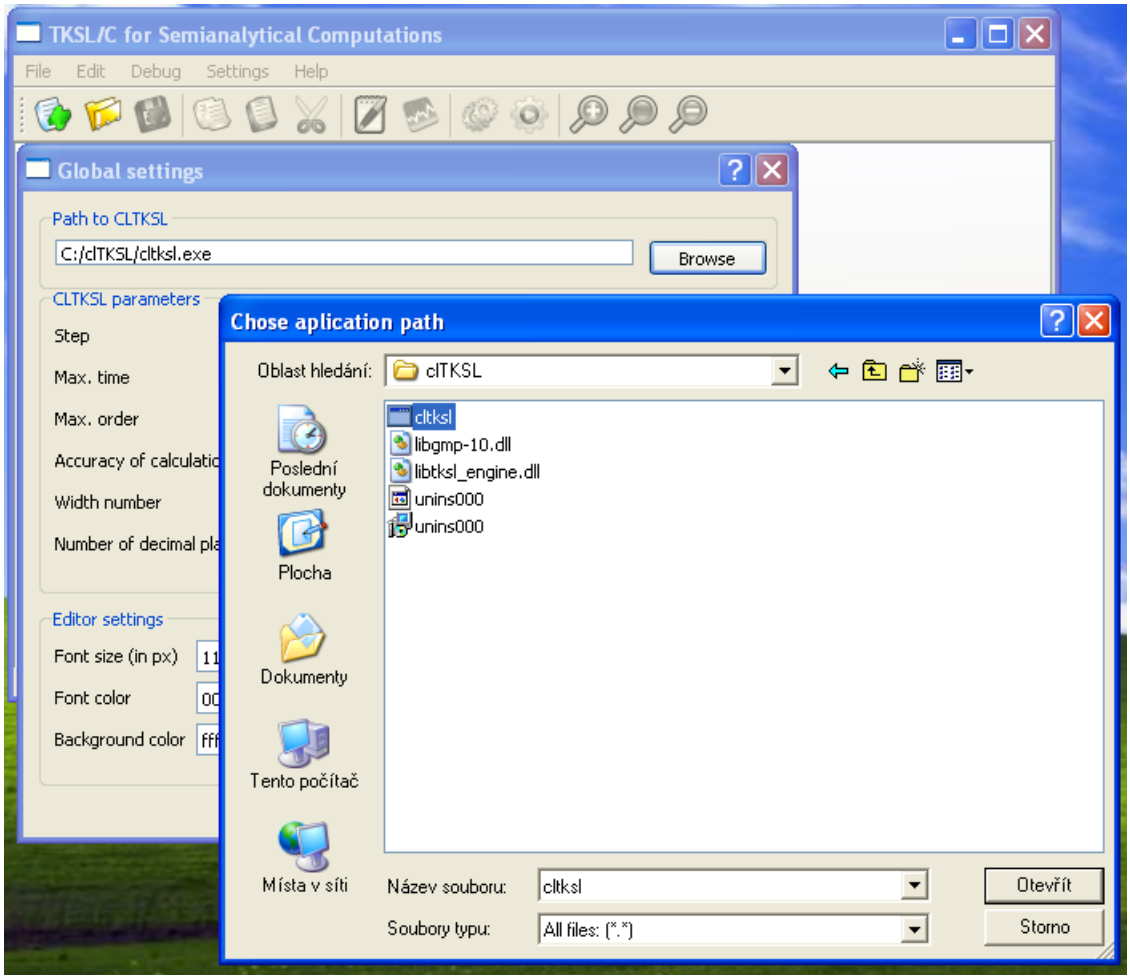
**info.txt** ... popis obsahuje média (obsahuje i tuto přílohu)
**bp.pdf** ... zpráva ve formátu PDF
**dotaznik.pdf** ... text dotazníků pro hodnocení rozhraní uživateli
**/examples/** ... adresář se zdrojovými kódy pro řešené úlohy
**/gui/source/** ... zdrojové kódy rozhraní přeložitelné pomocí Qt Creatoru
**/gui/win/** ... rozhraní přeložené na MS Windows 7 (spustitelné ve Win Xp i Vista)
**/gui/ubuntu/** ... binarní soubory rozhraní přeložené na Ubuntu 10.04 32bit
**/tex/** ... zdrojové kódy této dokumentace v LaTeXu
**/tkslc/** ... binární soubory TKSL pro MS Windows

# Příloha B

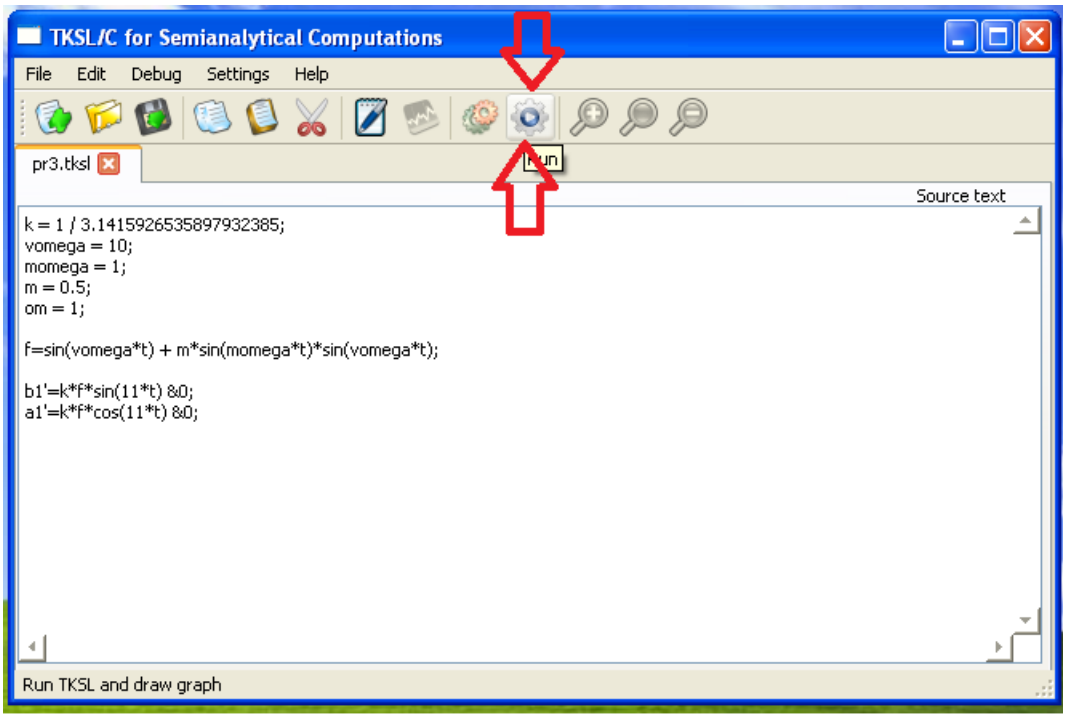
# Ukázka aplikace

## B.1 Spuštění rozhraní na MS Windows Xp

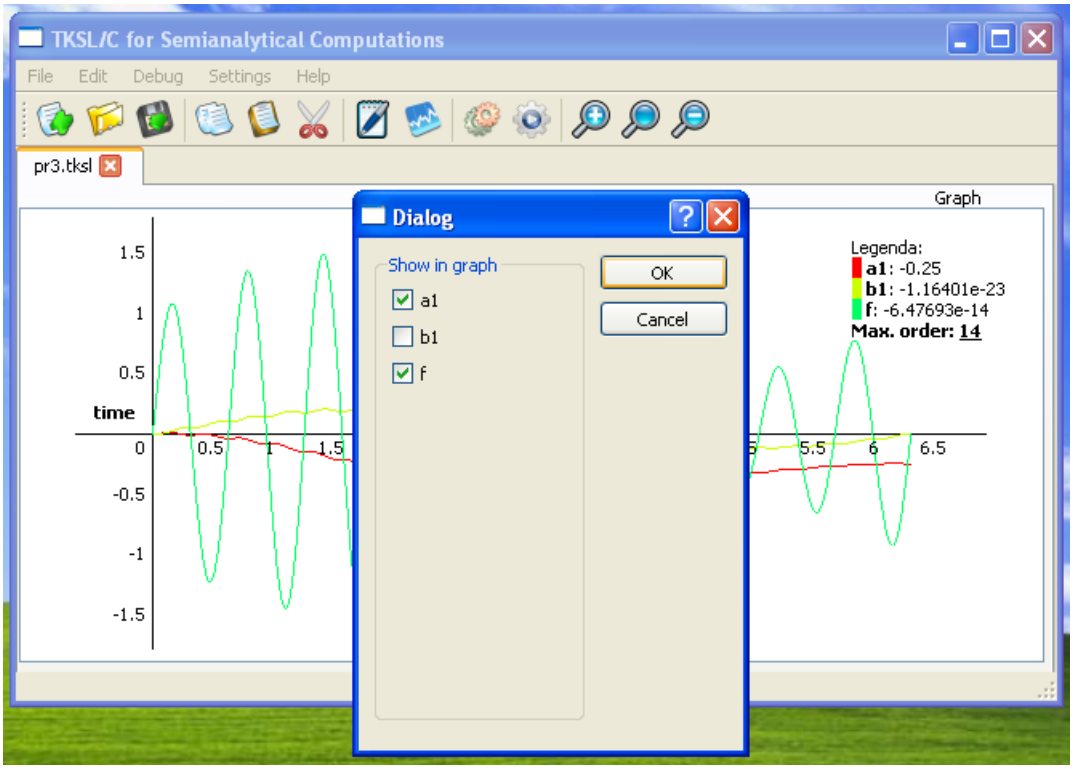
Obrázek B.1 ukazuje příklad nastavení cesty k TKSL/C po prvním spuštění.



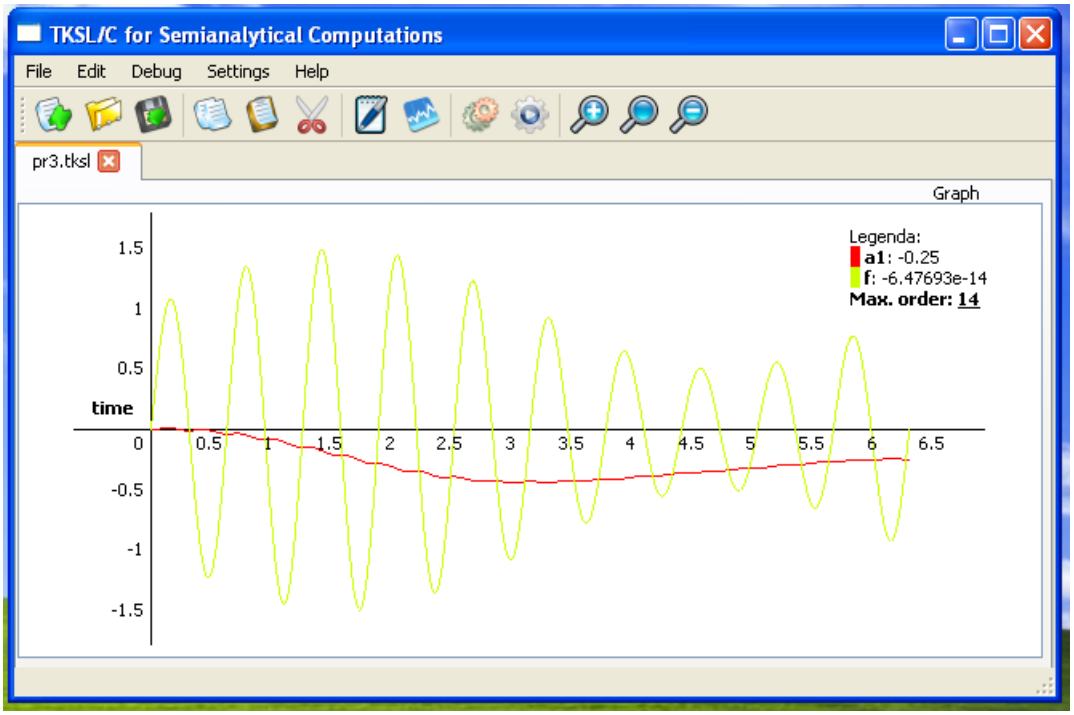

Obrázek B.1: Nastavení cesty k externímu CLTKSL.




Obrázek B.2: Tlačítko Run spouštící vykreslení grafu.

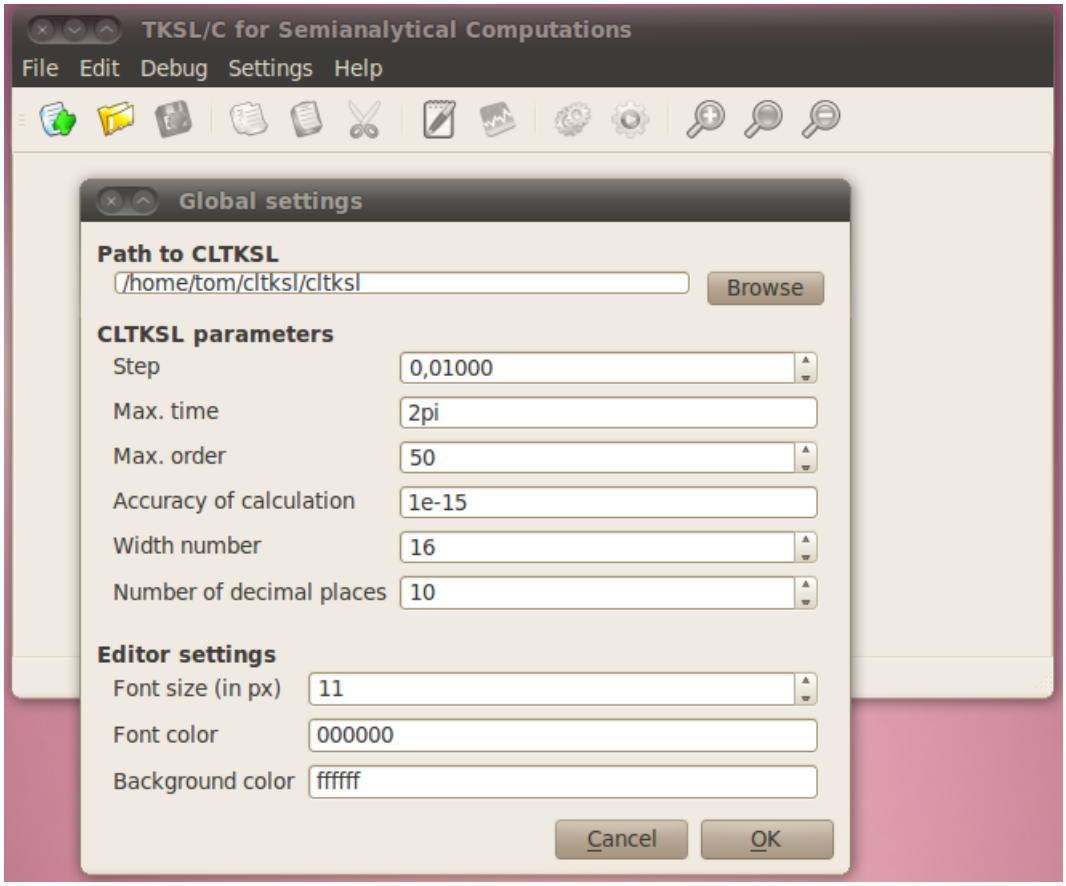



Obrázek B.3: Dialogové okno pro nastavení vykreslovaných křívek v grafu.

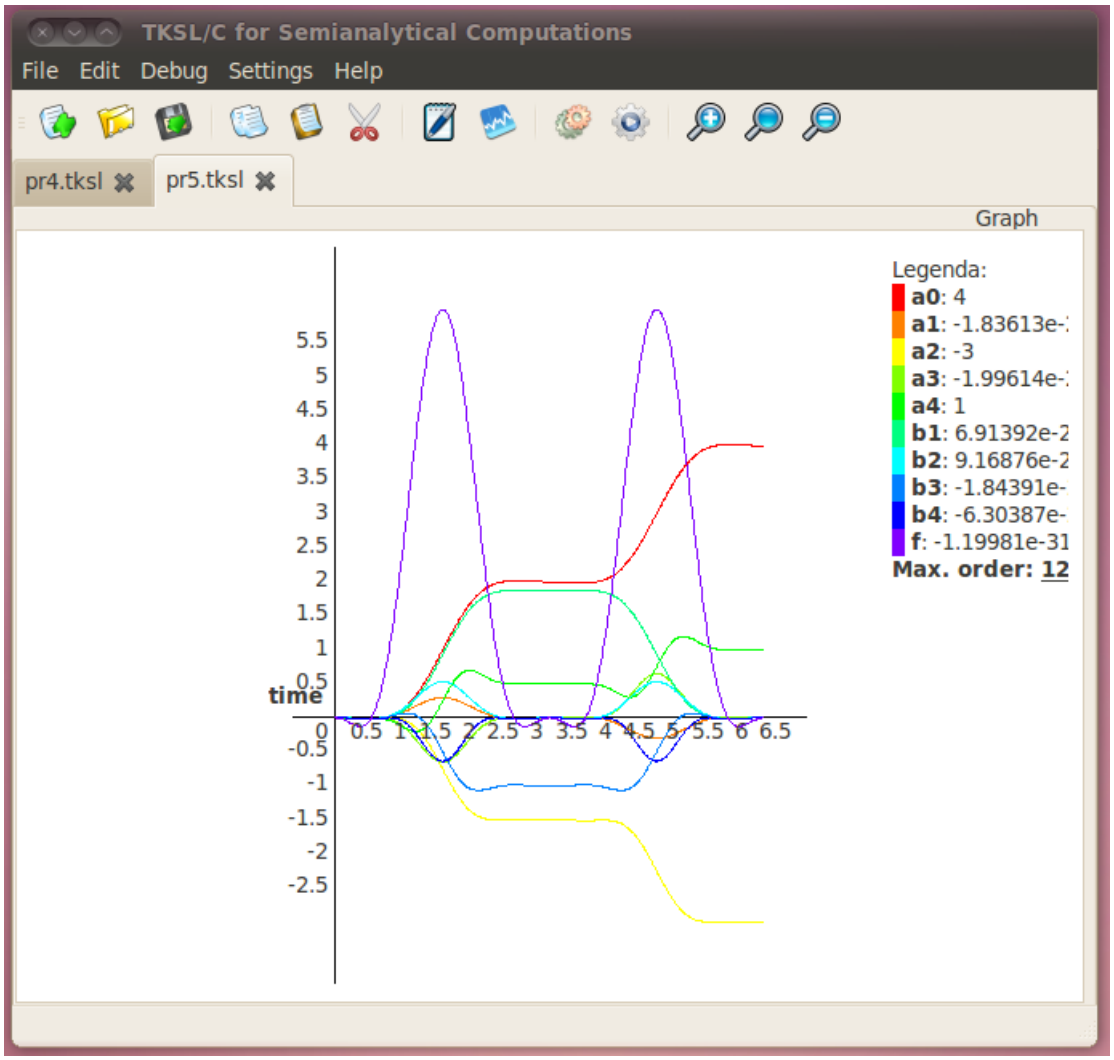



Obrázek B.4: Vykreslený graf.

## B.2 Ukázka spuštění rozhraní v Ubuntu 10.04




Obrázek B.5: Nastevení parametrů rozhraní včetně cesty k CLTKSL.




Obrázek B.6: Vykreslený graf v Ubuntu 10.04.